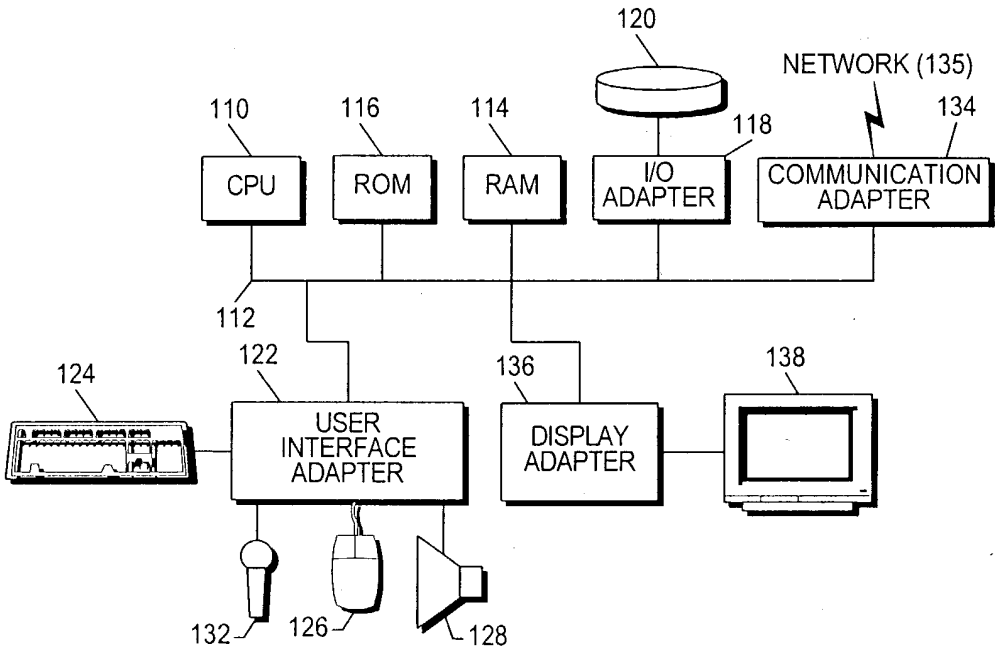

120
NETWORK (135)
110
116
114
118
134
CPU
ROM
RAM
I/O ADAPTER
COMMUNICATION ADAPTER
112
122
136
138
124
USER INTERFACE ADAPTER
DISPLAY ADAPTER
132
126
128

Fig. 1

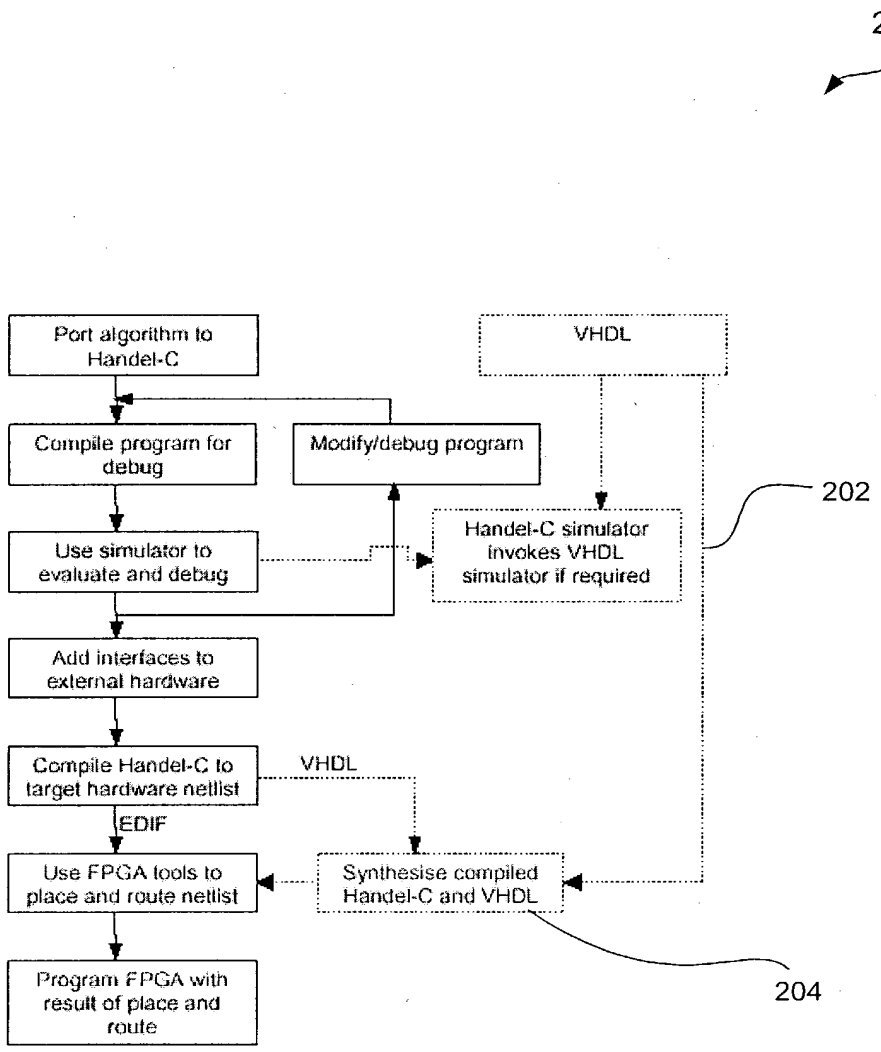

200
Port algorithm to Handel-C
VHDL
Compile program for debug
Modify/debug program
Use simulator to evaluate and debug
Handel-C simulator invokes VHDL simulator if required
202
Add interfaces to external hardware
Compile Handel-C to target hardware netlist
VHDL
EDIF
Use FPGA tools to place and route netlist
Synthesise compiled Handel-C and VHDL
204
Program FPGA with result of place and route

FIG. 2


| | |
|---|---|
| **Workspace window** | The area where you organise your projects. A project consists of all the files you need, plus information about what target you are compiling for. When you have assembled all the information that you need for a project, you can compile it. When you start Handel-C, the default position of the workspace window is on the left. |
| **Code editor** | Where you create and edit Handel-C source files. When you create or open a file, the default position of the code editor window is on the right. |
| **Output window** | When you compile a file, error messages and warnings are displayed in the output window. The default position of the output window is at the bottom of the screen. The output window has tabs for build messages and debug messages. |
| **Debug windows** | When your file has compiled, you can simulate it. The simulation steps the program through clock cycles, and allows you to look at the contents of any variables that are in scope. These are displayed in the **Variables** window.<br>You can select variables to display in the debug **Watch** window. The default position of the watch window is the bottom left-hand corner of the screen.<br>The call stack (the route by which you have called a function) is displayed in the **Call Stack** window.<br>You can see clock cycles in the **Clocks** window and current executing threads in the **Threads** window |

FIG. 3

400

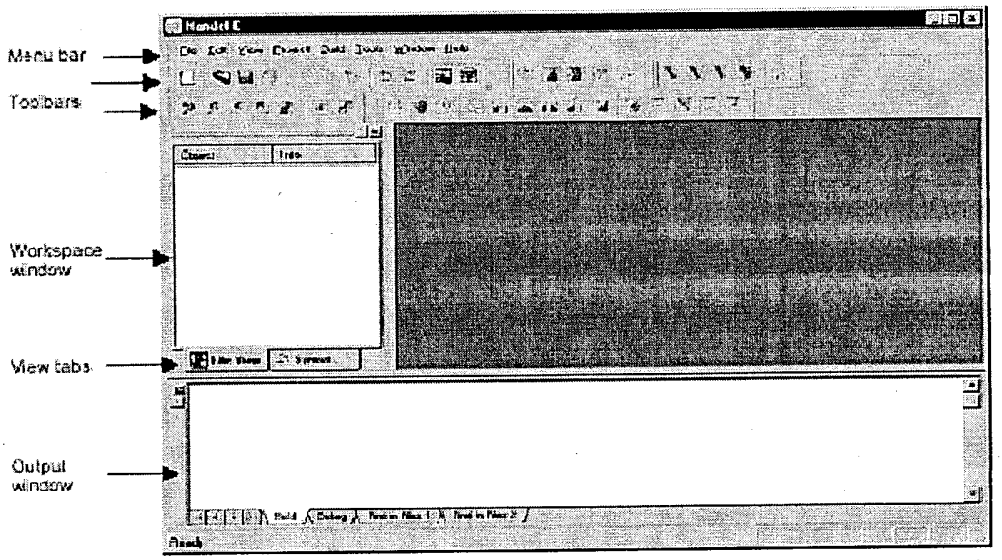

Menu bar
Toolbars
Workspace window
View tabs
Output window

FIG. 4

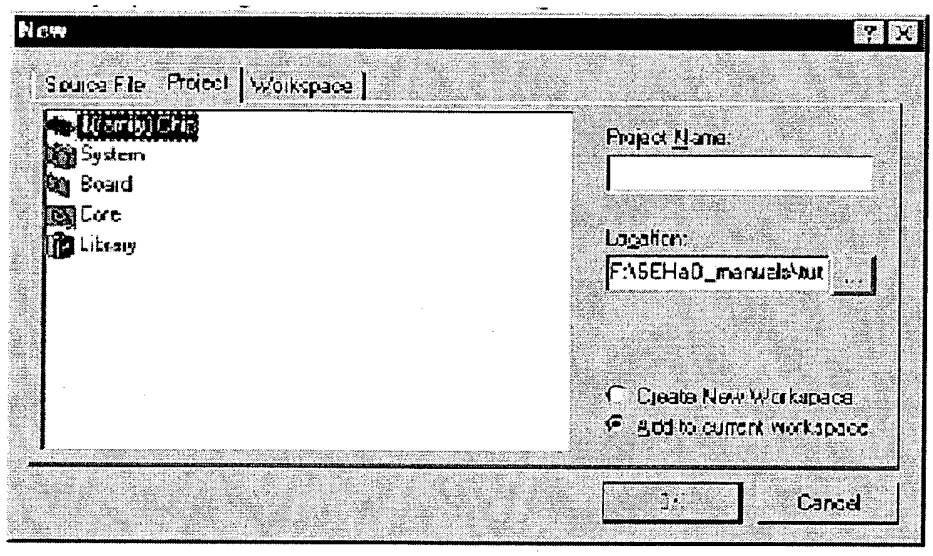

500
New
Source File
Project
Workspace
System
Board
Core
Library
Project Name:
Location:
F:\SEHaD_manuals\Aut
Create New Workspace
Add to current workspace
Cancel

FIG. 5

600

| | |
|---|---|
| a chip, system or board | Not targeted to a particular system or product |
| a core | A discrete piece of code, compiled to a specific architecture, that may be used as part of a larger design |
| a library | Pre-compiled Handel-C code that may be re-used or sold elsewhere |
| a pre-defined chip, system or board | Targeted to a known product. These systems will be optimised for that product, and should only be placed and routed onto that product. |

FIG. 6

700

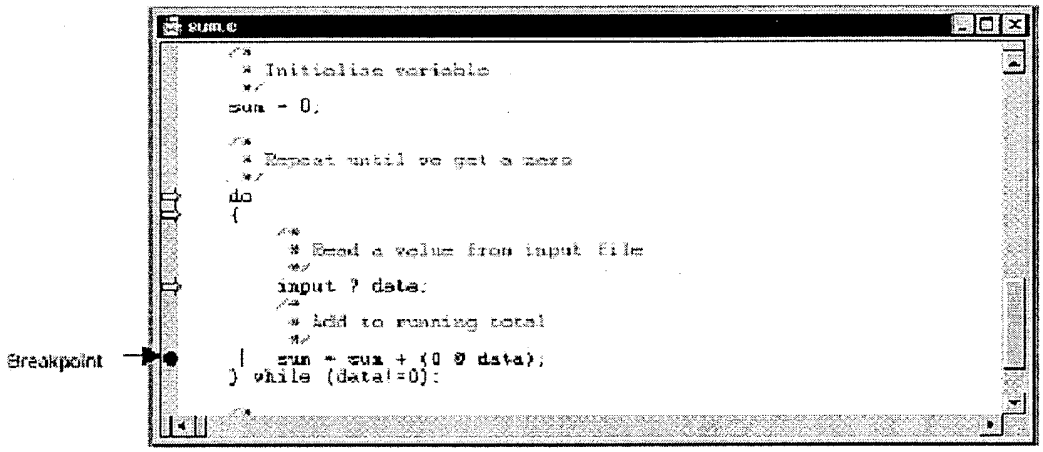

sum.c
/*
* Initialise variable
*/
sum = 0;
/*
* Repeat until we get a zero
*/
do
{
/*
* Read a value from input file
*/
input ? data:
/*
* Add to running total
*/
sum = sum + (0 @ data);
} while (data!=0);
Breakpoint

FIG. 7

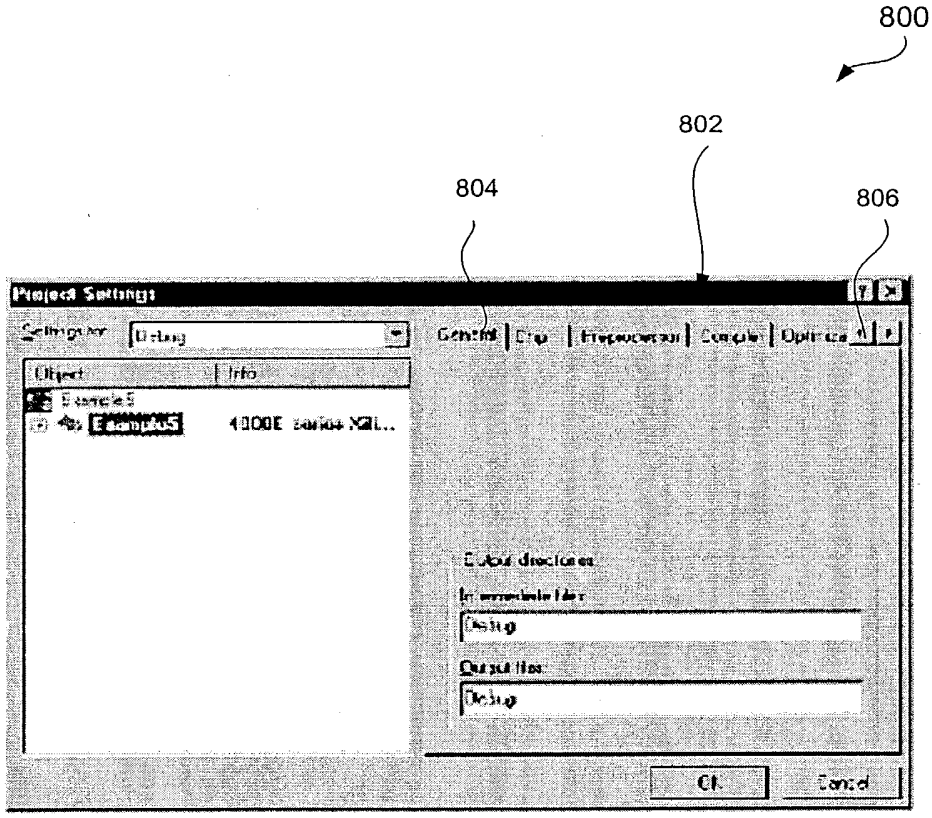

800
802
804
806
Project Settings
Debug
General
Info
Object
Output directories
Debug
Debug
OK
Cancel

FIG. 8

900

| Tab | Item | Meaning | Value | Default |
|---|---|---|---|---|
| **General** | | | | |
| | Intermediate files | The sub-directory where intermediate files are stored | Directory path name relative to the project directory | *configuration name* |
| | Output files | The sub-directory where the final output is stored (.dll, netlist etc.) | Directory path name relative to the project directory | *configuration name* |
| **Chip** | | | | |
| | Family | The family containing the part you are targeting | Select family from drop-down list | As required |
| | Part | The part number you are targeting | Type in part number | Depends on project |

FIG. 9A

900

| | | | | |
|---|---|---|---|---|
| **Preprocessor** | | | | |
| | Preprocessor definitions (-D) | Equivalent to the #define directive | Set as required | `DEBUG`, `NDEBUG`, `SIMULATE` |
| | Additional include directories (-I) | Add directories to the search path for include directories | Set as required | None |
| | Additional preprocessor options (-cpp) | Add any cpp commands | See the pre-processor manual | None |
| **Compiler** | | | | |
| | Generate debug information (-g) | Get the compiler to produce information for the debugger | Check for Yes | Checked |
| | Generate warning messages (-W) | Get the compiler to produce warning messages | Check for Yes | Checked |
| | Generate estimation information (-e) | Get the compiler to generate HTML files giving depth and timing information | Check for Yes | Unchecked |
| **Optimisations** | | | | |
| | Various levels of optimizations | Check as needed | Depends on target | |
| **Linker** | | | | |
| | Output format | Select the target for the compiler | Determined by target settings | As required |
| | Save browse info | Store information needed to browse symbols | Checked | Checked |
| | Additional Library Path | Directory path to search for libraries | Set as required | |
| | Command line for building simulator DLL (-cl) | Link options defined in the `cl.cf` file | Define how the C compiler is called to compile simulator .dll | Installed `cl.cf` settings. (Debug and Release only) |
| | Object/library modules | Specify modules to include in build | Set as required | |
| **Debugger** | | | | |
| | Working directory | Directory that the simulator uses as the current working directory | Directory path name relative to the project directory | Defaults to current project directory (.) |
| **Build commands** | | | | |
| | Compilation commands | Not yet implemented | | None |

FIG. 9B

900

| | | | | |
|---|---|---|---|---|
| | Output files | Not yet implemented | | None |

FIG. 9C

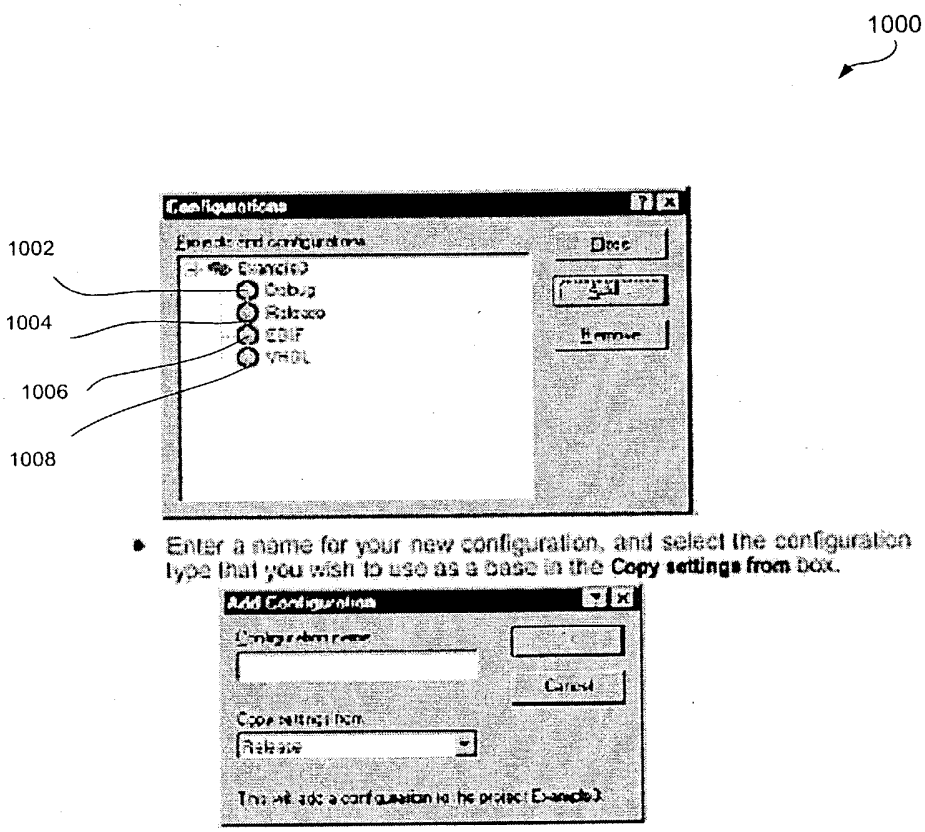

1000
1002
1004
1006
1008
Configurations
Debug
Release
EDIF
VHDL
Close
Remove
Enter a name for your new configuration, and select the configuration type that you wish to use as a base in the Copy settings from box.
Add Configuration
Cancel
Copy settings from:
Release

FIG. 10

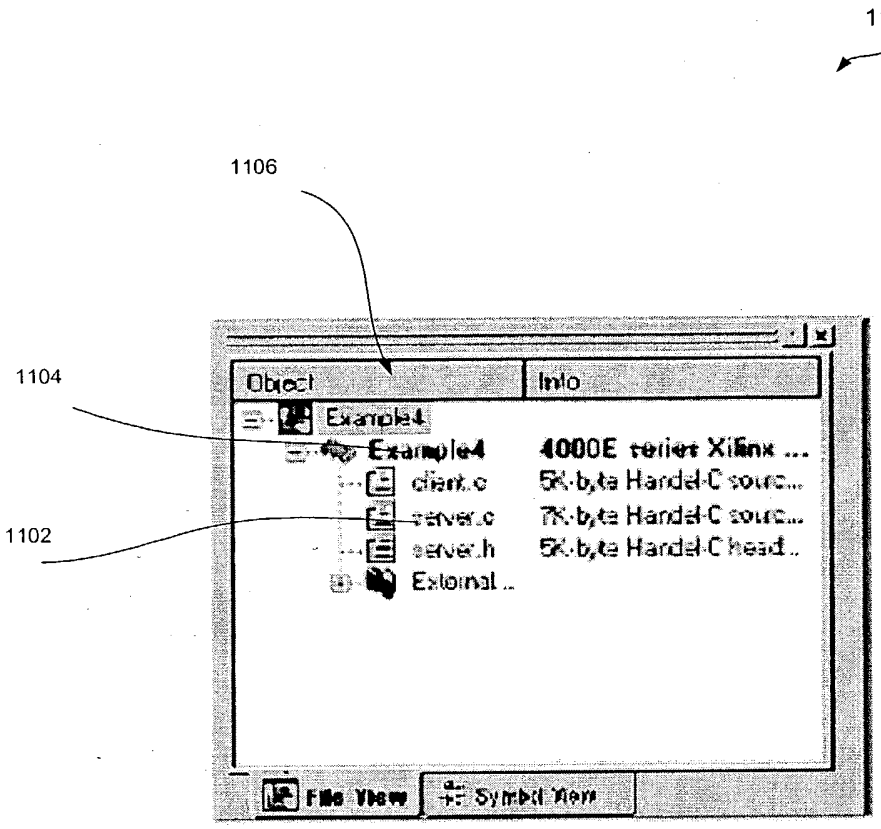

1100
1106
1104
1102
Object
Info
Example4
Example4
4000E series Xilinx ...
client.c
5K-byte Handel-C sourc...
server.c
7K-byte Handel-C sourc...
server.h
5K-byte Handel-C head...
External...
File View
Symbol View

FIG. 11

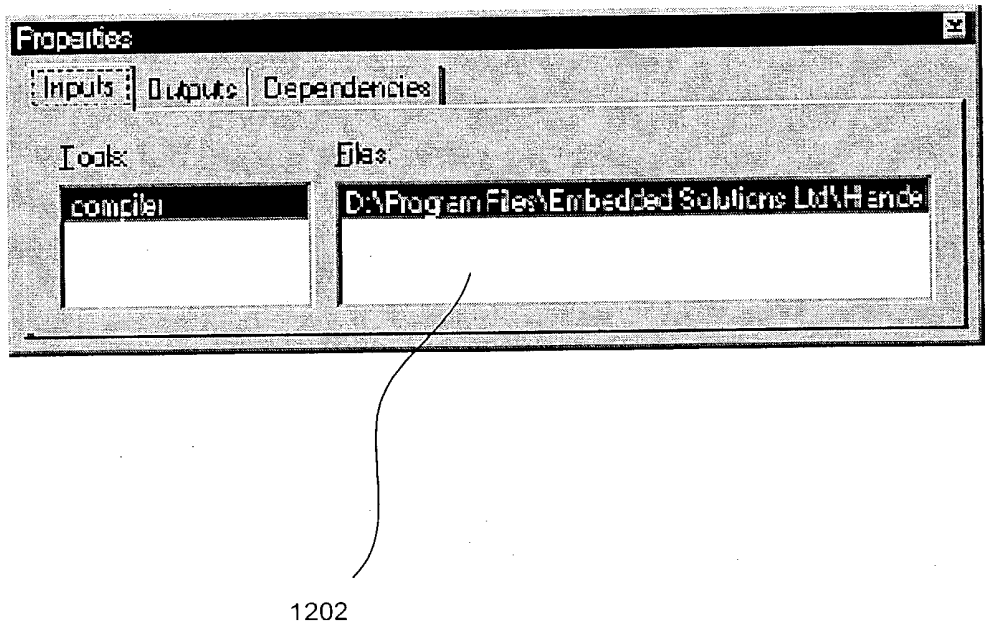
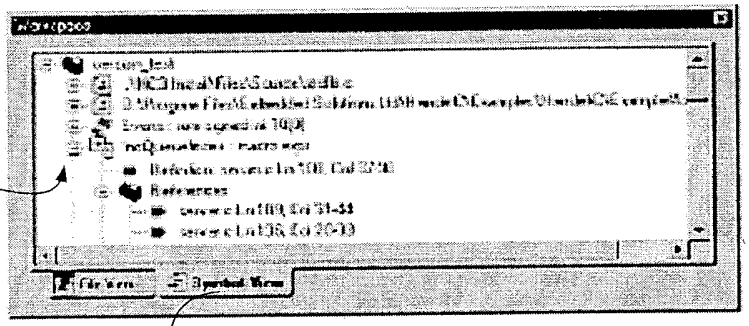

1200
Properties
Inputs
Outputs
Dependencies
Tools:
Files:
compiler
D:\Program Files\Embedded Solutions Ltd\Hiende
1202

FIG. 12

1300

| Icon | Meaning |
|---|---|
| | shared function, procedure or expression |
| | in-line function or macro |
| | variable |
| | memory (RAM, ROM, WOM or MPRAM) |
| | channel |
| | external interface |
| | signal |
| | Stacked positions that contain the related object (e.g. recursive macros) |
| | Position in the file containing the definition of the object |



1304

1302

FIG. 13

1400

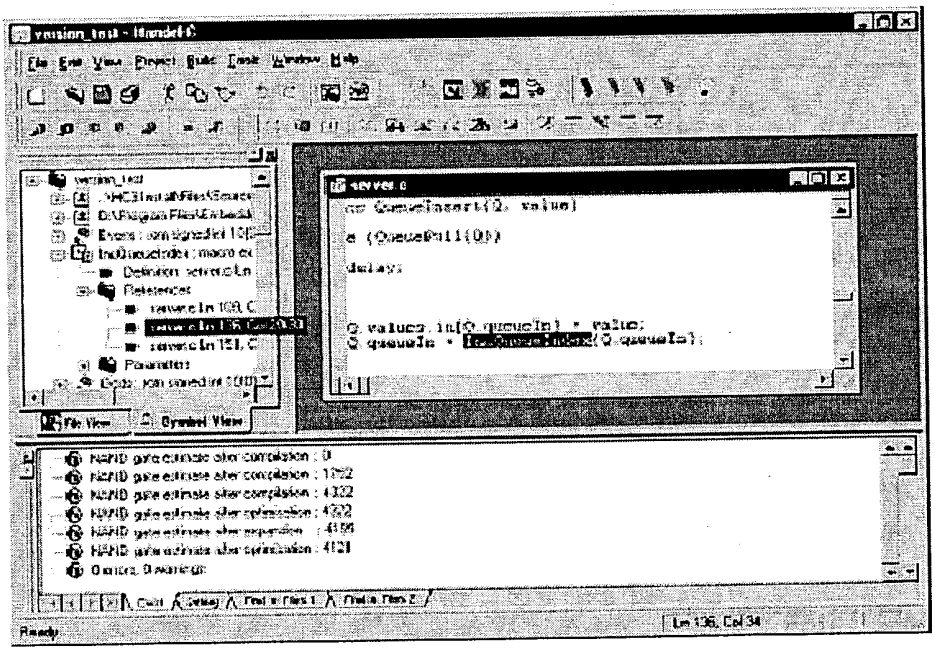


FIG. 14

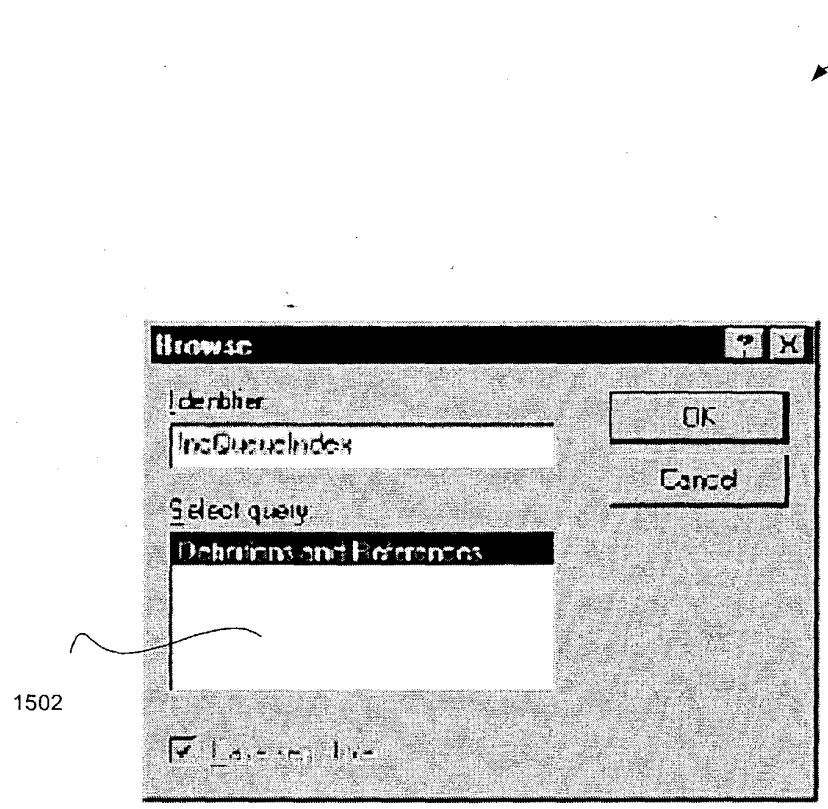

1500
Browse
Identifier
IncQueueIndex
OK
Cancel
Select query
Definitions and Parameters
1502

FIG. 15

1600

Browsing

| Button | Command | Function |
|---|---|---|
|  | **Go to Definition** | Jump to the source code line where the variable is defined |
|  | **Go to Reference** | Jump to the first source code line where the variable is used |
|  | **Previous Definition/Reference** | Jump to previous definition or reference |

FIG. 16A

1600

| | | |
|---|---|---|
| | **Next Definition/Reference** | Jump to next definition or reference |
| | **Pop context** | Returns you to your original position before you started browsing |

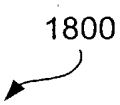
FIG. 16B

1700

| Command | Shortcut | Function |
|---|---|---|
| **Undo** | Ctrl+Z | Reverse a recent change to the active document or to the workspace |
| **Redo** | Ctrl+Y | Reverse a recent undo |
| **Cut** | Ctrl+X | Copy the current selection and delete it |
| **Copy** | Ctrl+C | Copy the current selection to the clipboard |
| **Paste** | Ctrl+V | Copy the clipboard to the current selection |
| **Delete** | Del | Delete the current selection |
| **Find** | Ctrl+F | Find a string or regular expression in the current file |
| **Find in files** | | Find a string or regular expression in selected files |
| **Replace** | Ctrl+H | Replace one string or regular expression with another in current file |

The **Edit** menu also has the **Bookmarks** and **Browse** sub-menus and the Breakpoints command.

FIG. 17




| Regular Expression | Description |
|---|---|
| (*x*) | The characters or expressions between the parentheses |
| . | (Period.) Any single character. |
| ^ | Start of line. |
| $ | End of line. |
| \t | Tab character |
| *x*\|*y* | A match for either *x* or *y*. For example, `a(team\|class)` will match either **ateam** or **aclass**. |
| *x** | Zero, one or many copies of *x*. For example, `ba*c` matches **bac**, **baac**, **baaac** and **bc**. |
| *x*? | None or one *x*. For example, `ba?c` matches **bac** or **bc**. |
| *x*+ | At least one or more of *x*. For example, `ba+c` matches **bac**, **baac**, **baaac**, but not **bc**. |
| [*xyz*]<br>[*x-y*] | Matches one character from the set in the brackets. Use a dash (-) to include all characters in a range (e.g., `[_A-Za-z]` matches an underscore or any letter, and `[_A-Za-z][_A-Za-z0-9]*` matches an alphanumeric string that can include underscores). Use `[xyz-]` or `[-xyz]` if you want to include a dash in the set. If you need a ] in the set use `[]xyz]`. |
| [^*xyz*] | Matches one character that is not in the brackets. For example, `x[^0-9]` matches **xa**, but not **x0** or **x2**. |
| *x* | Matches the character *x*, even if *x* is one of the magic characters `^$[].*+?` listed above. For example, `^pig` matches pig at the start of a line, but `\^pig` matches the string `^pig` anywhere on a line. |

FIG. 18

1900

| Directory | File | File type |
|---|---|---|
| Where saved | prog.c | Source file |
| Workspace directory | prog.hw | Workspace |
| Project directory | example1.hp | Your project file |
| Output directory | example1.dll | Part of the simulator |
| | example1.lib | Part of the simulator |
| | example1.hb | The project browse file for the symbol view window |
| | example1.exp | Part of the simulator |
| | prog.hb | A program browse file used for symbol view |
| Intermediate directory | prog.obj | Handel-C object file built during compilation. |

FIG. 19

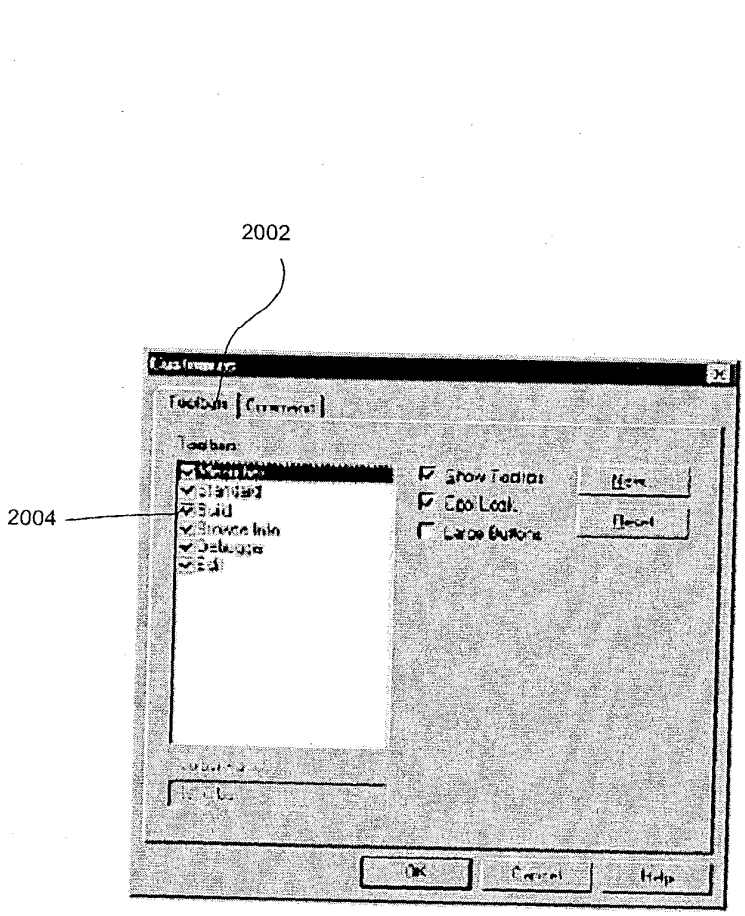

2000
2002
2004
Toolbars
Commands
Toolbars:
Show Tooltips
Cool Look
Large Buttons
New...
Reset
OK
Cancel
Help

FIG. 20

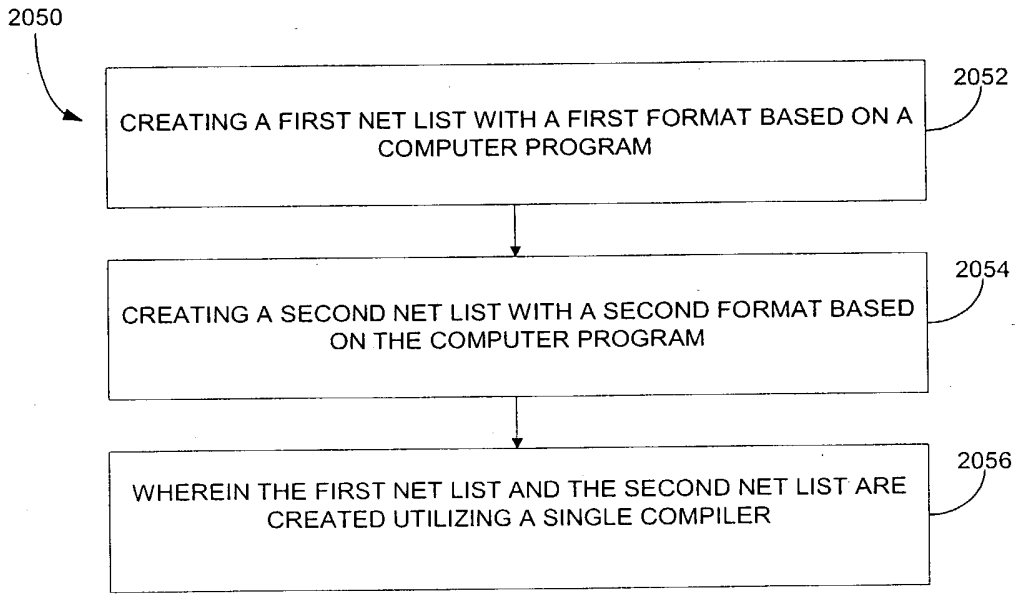
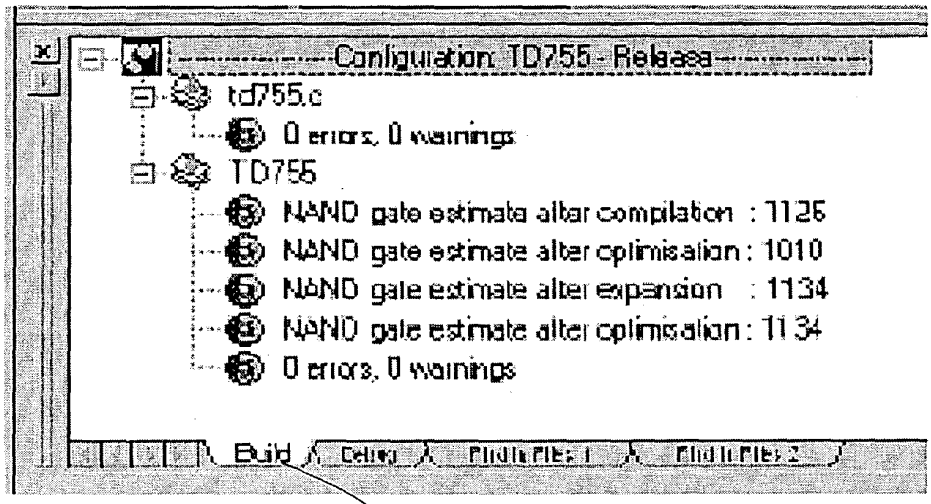

2050
CREATING A FIRST NET LIST WITH A FIRST FORMAT BASED ON A COMPUTER PROGRAM
2052
CREATING A SECOND NET LIST WITH A SECOND FORMAT BASED ON THE COMPUTER PROGRAM
2054
WHEREIN THE FIRST NET LIST AND THE SECOND NET LIST ARE CREATED UTILIZING A SINGLE COMPILER
2056

Fig. 20A

2100




2102

FIG. 21

2200

| Command | Shortcut | Function |
|---|---|---|
| **Compile** | Ctrl+F7 | Run the compiler on the active document (which must be a `.c` file), to generate its `.obj` file. |
| **Build** *project* | F7 | Build this project: Run the compiler on all `.c` files that are newer than their `.obj` files, then run the linker on the `.obj` files to make the `.dll`, EDIF or VHDL file. |
| **Rebuild All** | | Rebuild all the files in this project: like Build, except that all `.c` files are compiled. |
| **Clean** | | Delete all the files that are created by Build. |
| **Start Debug** | | Pop-up menu |
| **Go** | F5 | (Build project if not built.) Run the simulator at full speed (until a breakpoint or other stop is reached) |
| **Step Into** | F11 | (Build project if not built.) Run to the first statement in the function or macro invoked in the current line. If the current line is not a function or macro invocation, run to the next statement. |
| **Run to Cursor** | Ctrl+F1 | (Build project if not built.) Run up to the line containing the text cursor. |
| **Set Active Configuration** | | Shows a dialogue box where the user can choose the active configuration |
| **Configurations...** | | Shows a dialogue box where the user can add, remove or edit configurations. |

FIG. 22

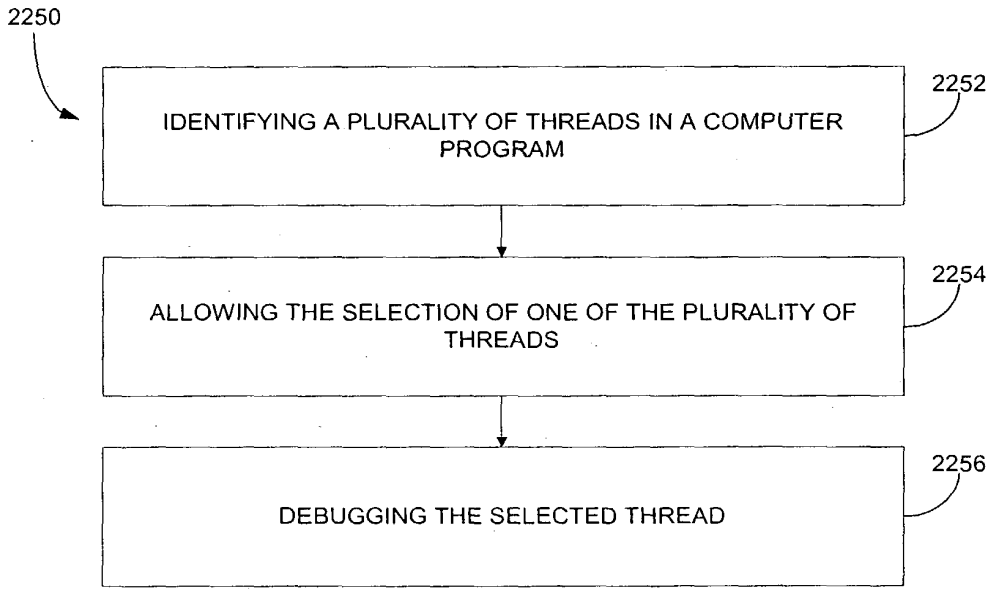
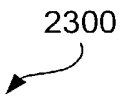

2250
IDENTIFYING A PLURALITY OF THREADS IN A COMPUTER PROGRAM
2252
ALLOWING THE SELECTION OF ONE OF THE PLURALITY OF THREADS
2254
DEBUGGING THE SELECTED THREAD
2256

Fig. 22A

2300

| Command | Shortcut | Function |
|---|---|---|
| **Go** | F5 | Run the simulator at full speed (until a breakpoint or other stop) |
| **Restart** | Ctrl+Shift+F5 | Run the simulator, starting at the first line of the program. |
| **Stop Debugging** | Shift+F5 | Stop the simulation |
| **Break** | | Pause the simulation (when it is running) |
| **Show Next Statement** | ALT + keypad * | Display the statement in the current thread that will be executed on the next clock cycle |
| **Step Into** | F11 | Run to the first statement in the function invoked in the current line. If the current line is not a function invocation, run to the next statement on a complete clock cycle. |
| **Step Over** | F10 | Run until the start of the next statement (step over a function) |
| **Step Out** | Shift+F11 | Run until the start of the statement after the line which invoked the current function (step out of a function) |

FIG. 23A




| | | |
|---|---|---|
| **Run to Cursor** | Ctrl+F10 | Run until the line containing the text cursor is reached. |
| **Advance** | Ctrl+F11 | Advance a partial clock cycle, to the next code line |

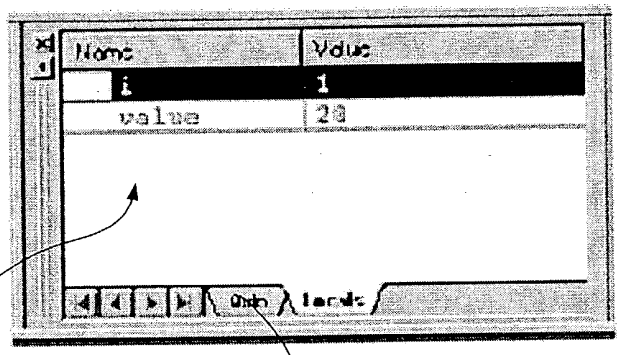
FIG. 23B

2400

| Window | Shortcut to open | Function |
|---|---|---|
| Editor window | Appears by default | The code editor window for the source code that you are debugging, marked to show the current execution points and breakpoints. |
| **Watch** | Alt+9 | A window with four tabs, in which you can view the contents of selected variables. |
| **Call Stack** | Alt+7 | A window that shows the calling path to the function you are in. |
| **Variables** | Alt+4 | Variables window. |
| **Clocks** | Alt+9 | Displays all current clocks and their values. |
| **Threads** | **Alt+5** | Displays all current threads with a unique identifier, and allows you to select one to view. |

FIG. 24

2500




2502

2504

FIG. 25

2600

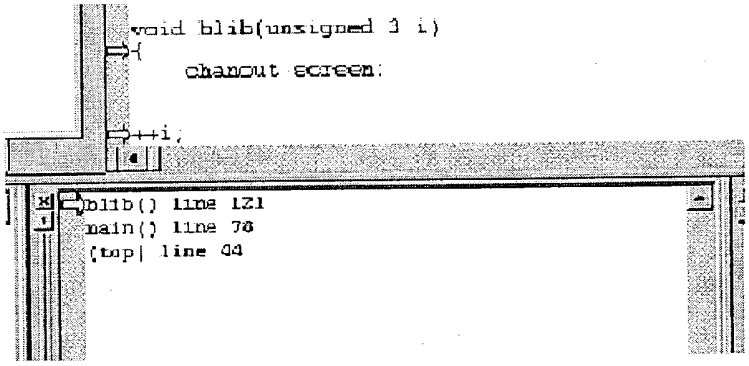

void blib(unsigned i)
{
chamout screen;
++i;
blib() line 121
main() line 78
[top] line 44

FIG. 26

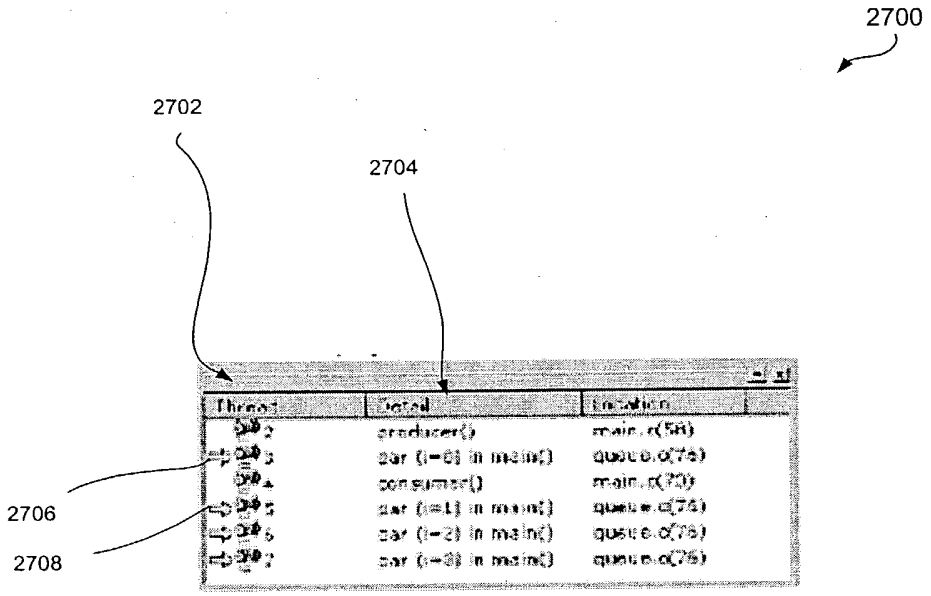



FIG. 27

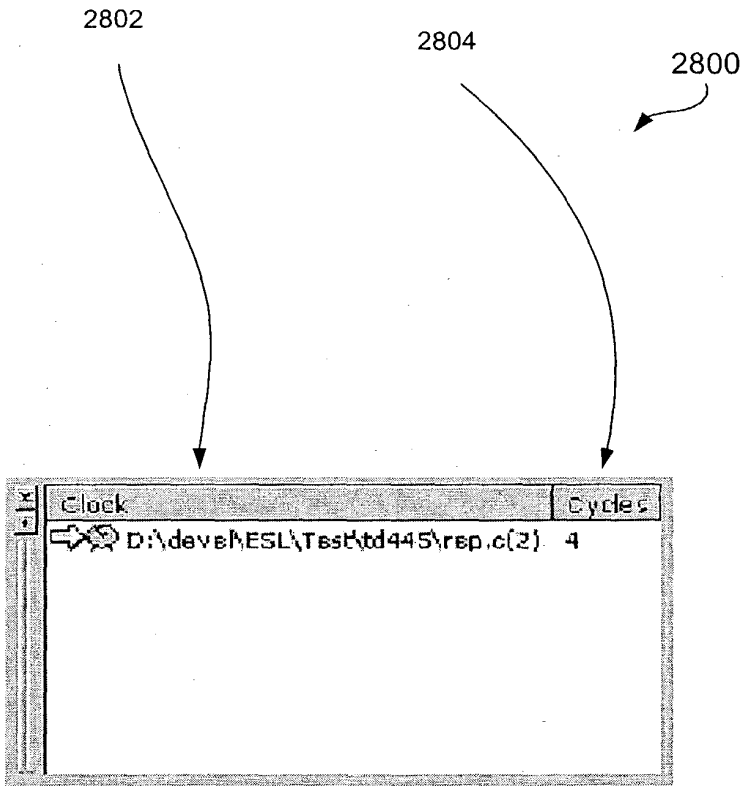



FIG. 28

2900

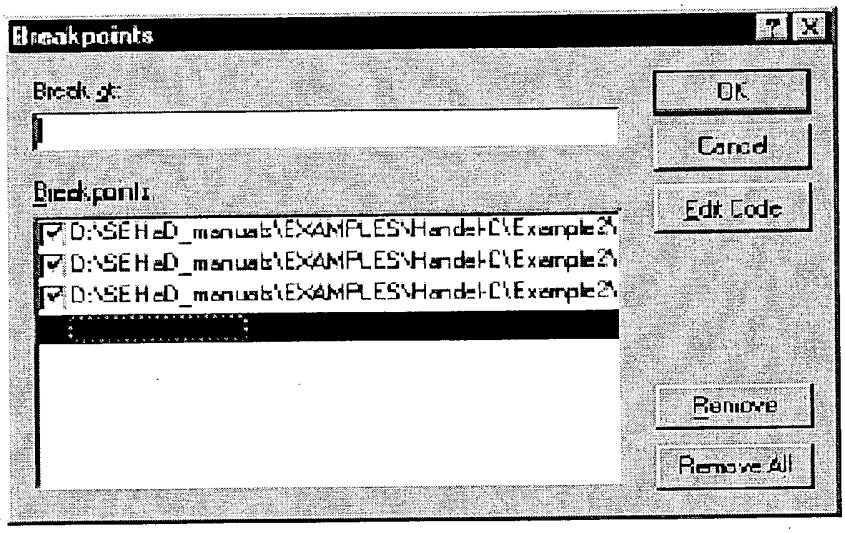

Breakpoints
Break at:
Breakpoints:
D:\SEHeD_manuals\EXAMPLES\Handel-C\Example2\
D:\SEHeD_manuals\EXAMPLES\Handel-C\Example2\
D:\SEHeD_manuals\EXAMPLES\Handel-C\Example2\
OK
Cancel
Edit Code
Remove
Remove All

FIG. 29

3100

| | |
|---|---|
| Strong typing | Handel-C has variables which can be defined to be of any width. |
| | Casting can't change width. |
| | There are no automatic conversions between signed and unsigned values. Instead, values must be 'cast' between types to ensure that the programmer is aware that a conversion is occurring that may alter the meaning of a value. |
| | Pointers can only be cast to `void` and back, between `signed` and `unsigned` and between similar `structs`. You cannot cast pointers to any other type |
| True parallelism | You can have multiple main functions in a project. Each Handel-C main function must be associated with a clock. |
| | Although implicitly sequential, Handel-C has parallel constructs which allow you to speed up your code |
| Width of variables | Handel-C has variables which can be defined to be of any width. |
| | In ISO-C, bit fields are made up of words, and only the specified bits are accessed, the rest are padded. Since there are no words in Handel-C, no form of packing can be assumed. |
| | If you have an `array[4]` and you use its index as a counter, the index width will be assumed by the Handel-C compiler to be two bits wide (to hold the values 0 – 3). It will not be able to hold the value 4. |
| No side-effects allowed | Instead of writing complex single statements, it is more efficient in Handel-C to write multiple single statements and run them in parallel |
| | You cannot perform two assignments in one statement. |
| | `auto` variables cannot be initialised, as that means that hidden clock cycles are required. Instead, they must be explicitly assigned to in a separate statement. |

FIG. 30

3100

| | |
|---|---|
| | You cannot have empty loops in Handel-C |
| Constrained functions | Functions may not be recursive. |
| | Variable length parameter lists are not supported. |
| | Old-style function declarations are not supported. |

FIG. 31

3200

| In Both | In Conventional C Only | In Handel-C Only |
|---|---|---|
| int | double | chan |
| unsigned | float | ram |
| char | union | rom |
| long | | wom |
| short | | mpram |
| enum | | signal |
| register | | chanin |
| static | | chanout |
| extern | | undefined |
| struct | | interface |
| volatile | | <> |
| void | | inline |
| const | | typeof |
| auto | | |
| signed | | |
| typedef | | |
| void | | |
| volatile | | |

FIG. 32

3300

| In Both | In Conventional C Only | In Handel-C Only |
|---|---|---|
| {;} | | par |
| switch | | delay |
| do … while | | ? |
| while | | ! |
| if … else | | prialt |
| for (;;) | | seq |
| break | | ifselect |
| continue | | |
| return | | |
| goto | | |
| assert | | |

FIG. 33

3400

| In Both | In Conventional C Only | In Handel-C Only |
|---|---|---|
| * (pointer indirection) | sizeof | select(...) |
| & (address of) | | width(...) |
| - | | @ |
| + | | \\ |
| * (multiplication) | | <- |
| / | | [:] |
| | | let...in |
| % | | |
| << | | |
| >> | | |
| > | | |
| < | | |
| >= | | |
| <= | | |
| == | | |
| != | | |
| & (bitwise and) | | |
| ^ | | |
| \| | | |
| ? : | | |
| [] | | |
| ! | | |
| && | | |
| ~ | | |
| \|\| | | |
| -> | | |

FIG. 34

3500

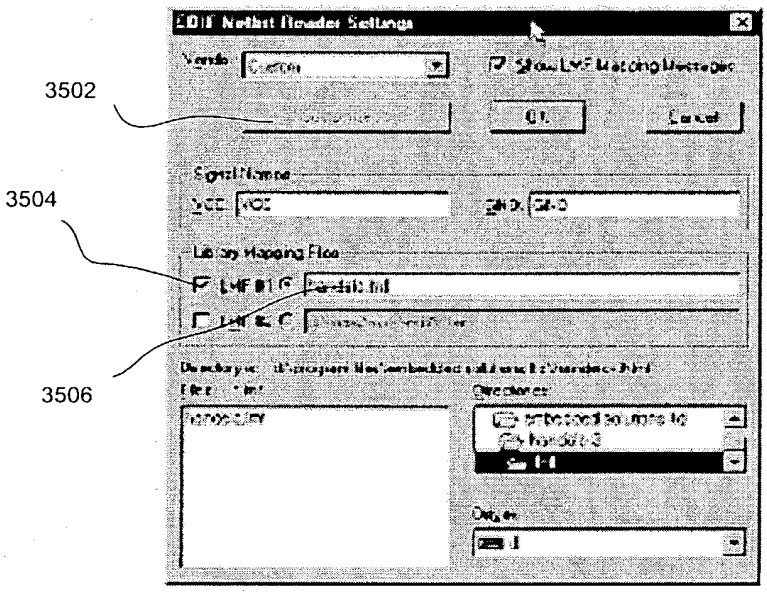



FIG. 35

3600

3602

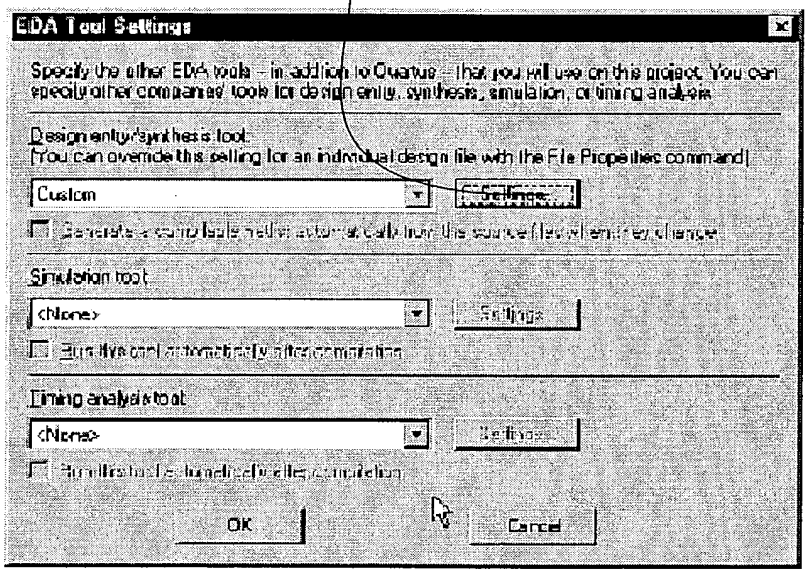
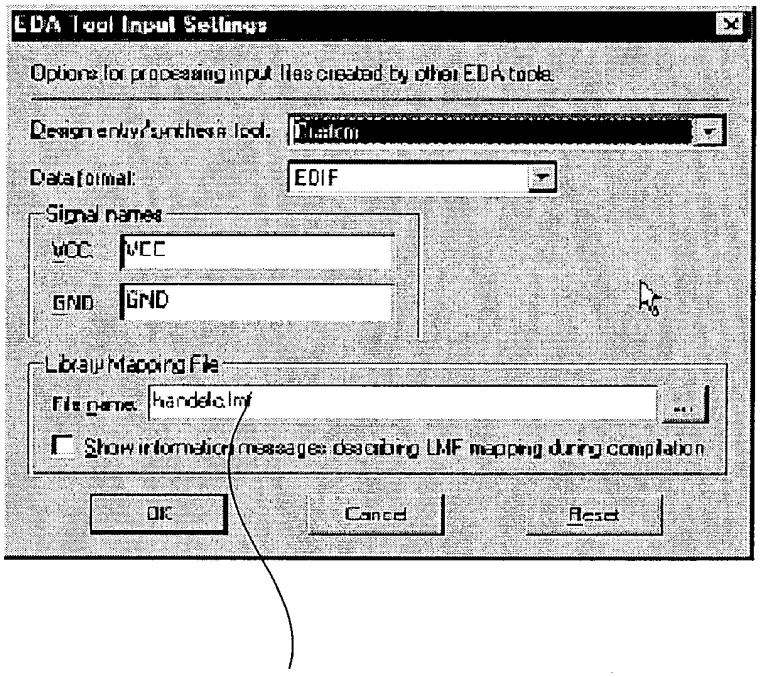



FIG. 36

3700




3702

FIG. 37

3800

```
interface port_in(int 4 signals_to_HC with
                        {busformat="B[1]} read();
```

would produce wires

```
signals_to_HC[0]
signals_to_HC[1]
signals_to_HC[2]
signals_to_HC[3]
```

FIG. 38

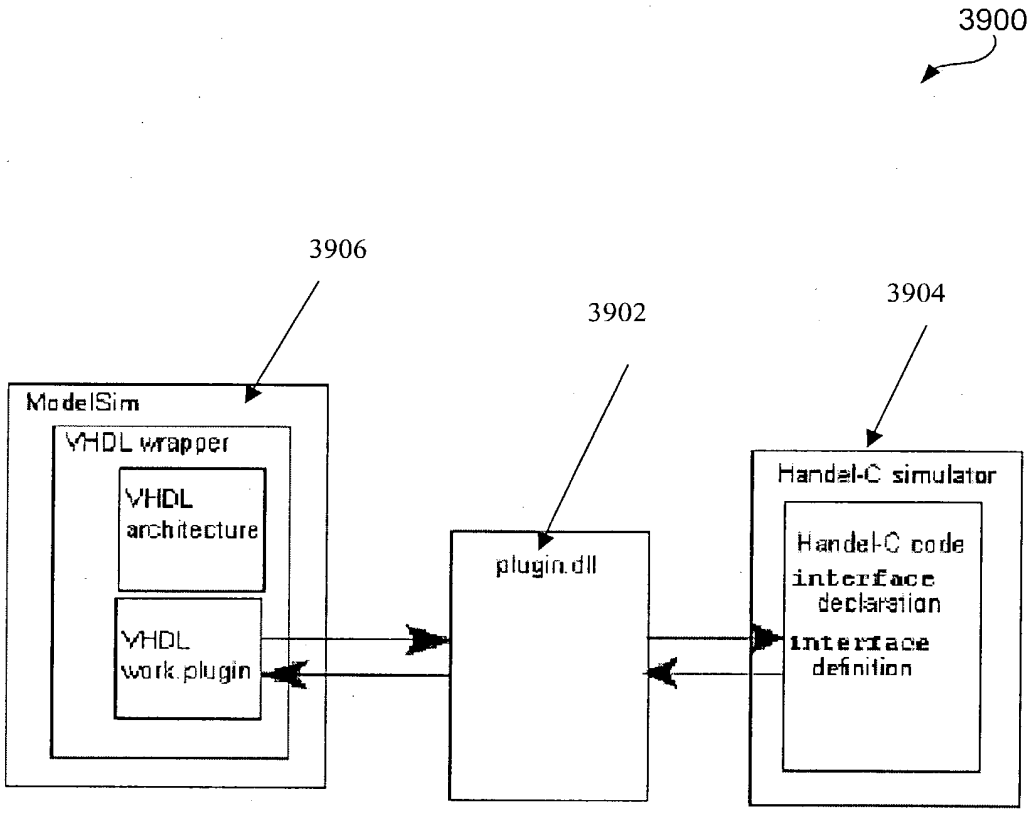

3900
3906
3902
3904
ModelSim
VHDL wrapper
VHDL architecture
VHDL work.plugin
plugin.dll
Handel-C simulator
Handel-C code
interface declaration
interface definition

FIG. 39

4000

| Keyword | Function | Default |
|---|---|---|
| base | specify display base for variables in debugger | 10 |
| extlib | specify external plugin for simulator | None |
| extfunc | specify external simulator function for this port | PlugInSet or PlugInGet |
| extpath | specify any direct logic (combinatorial logic) connections to another port | None |
| extinst | specify connection to external code | None |

FIG. 40A


| | | |
|---|---|---|
| warn | disable some compiler warnings | No |

FIG. 40B

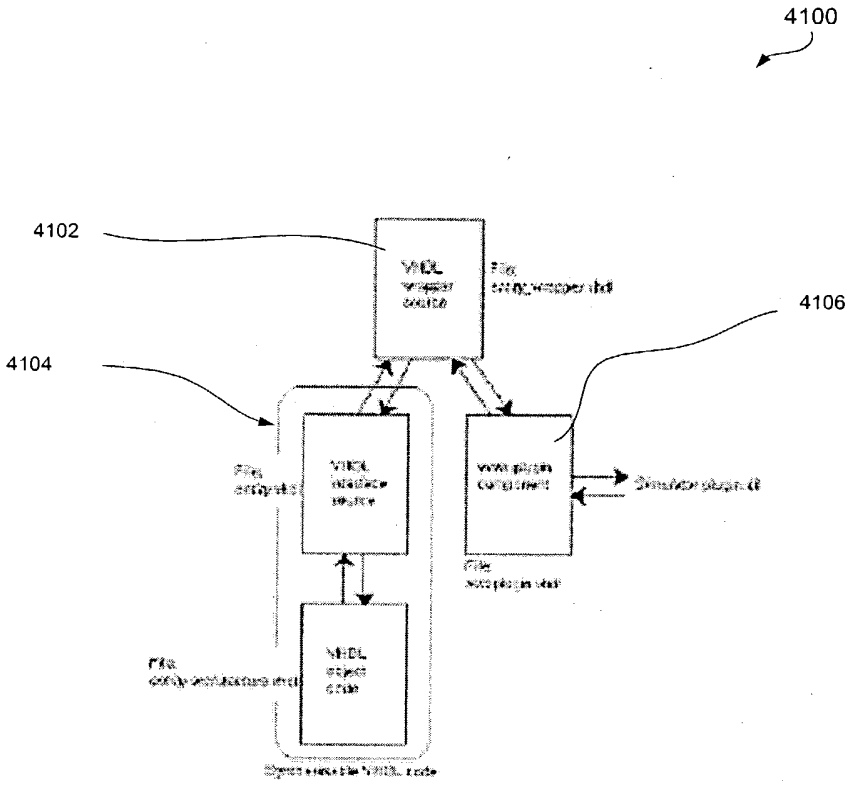



FIG. 41

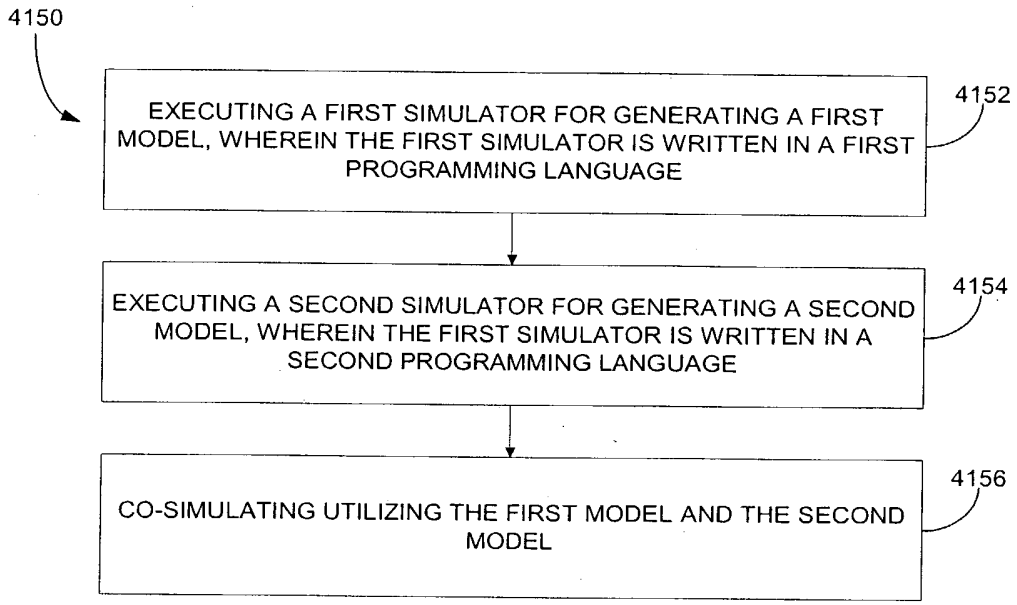

4150
EXECUTING A FIRST SIMULATOR FOR GENERATING A FIRST MODEL, WHEREIN THE FIRST SIMULATOR IS WRITTEN IN A FIRST PROGRAMMING LANGUAGE
4152
EXECUTING A SECOND SIMULATOR FOR GENERATING A SECOND MODEL, WHEREIN THE FIRST SIMULATOR IS WRITTEN IN A SECOND PROGRAMMING LANGUAGE
4154
CO-SIMULATING UTILIZING THE FIRST MODEL AND THE SECOND MODEL
4156

Fig. 41A

4200

| When used | Function call | How often |
|---|---|---|
| First use of simulator in Handel-C session | `PlugInOpen` | once per plugin |
| Start of simulation | `PlugInOpenPort` | once per interface port using the plugin |

FIG. 42A

4200

| | | |
|---|---|---|
| | PlugInOpenInstance | once per instance (copy) of plugin |
| Simulator data transfer | PlugInSet | called when data on a port sending data TO the plugin changes |
| | PlugInGet | called whenever the simulator wishes to read data FROM the plugin |
| Start of simulated clock cycle | PlugInStartCycle | |
| Middle of cycle | PlugInMiddleCycle | called immediately before the simulator variables are updated |
| End of cycle | PlugInEndCycle | |
| End of simulation | PlugInClosePort | once per interface port using the plugin |
| | PlugInCloseInstance | once per instance of the plugin |
| End of Handel-C session | PlugInClose | once per plugin |

FIG. 42B

4300

| | Possible Values | Default | Meaning |
|---|---|---|---|
| **extlib** | Name of a plugin **.dll** | None | Specify external plugin for simulator |
| **extfunc** | Name of a function within the plugin | **PlugInSet** or **PlugInGet** depending on port direction | Specify external function within the simulator for this port |
| **extinst** | Instance name (with optional parameters) | None | Specify simulation instance used |

FIG. 43

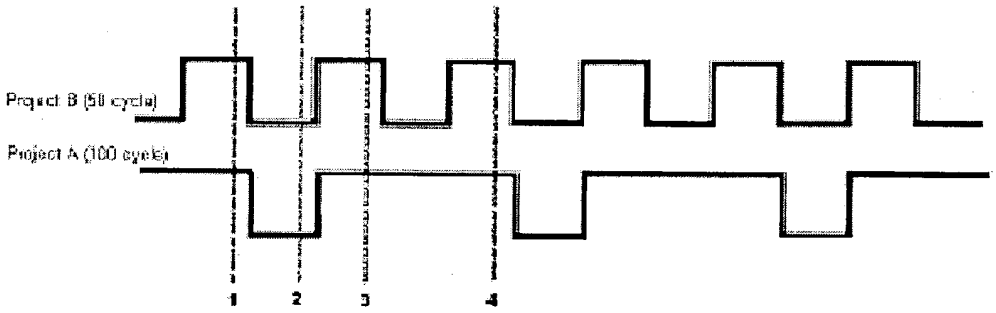



FIG. 44

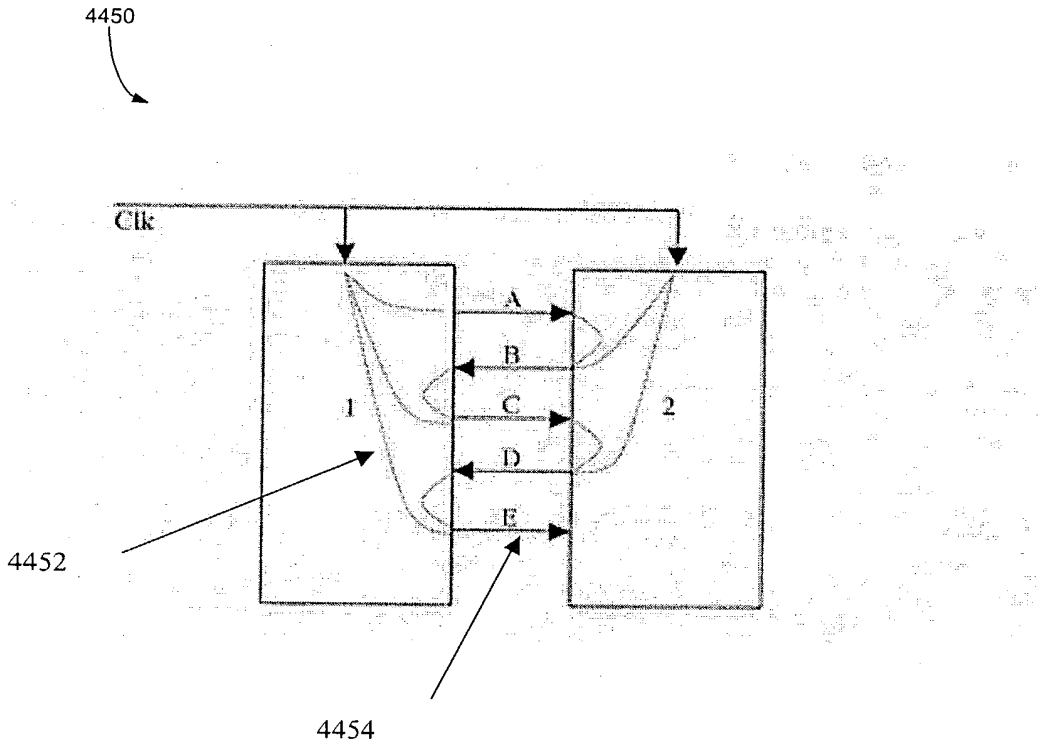



Fig. 44A

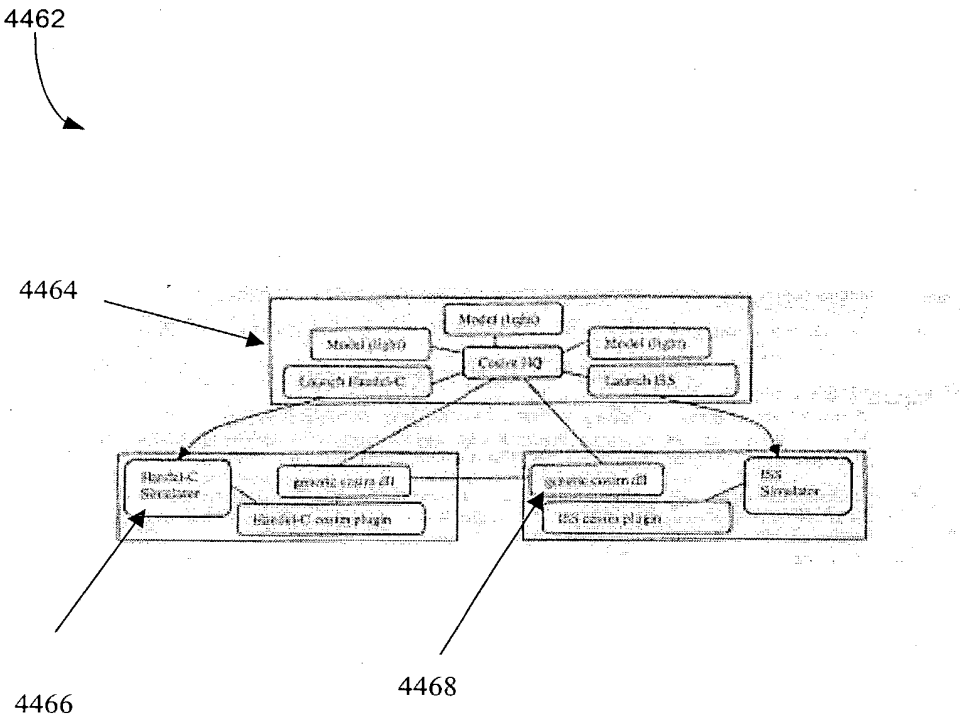



Fig. 44B

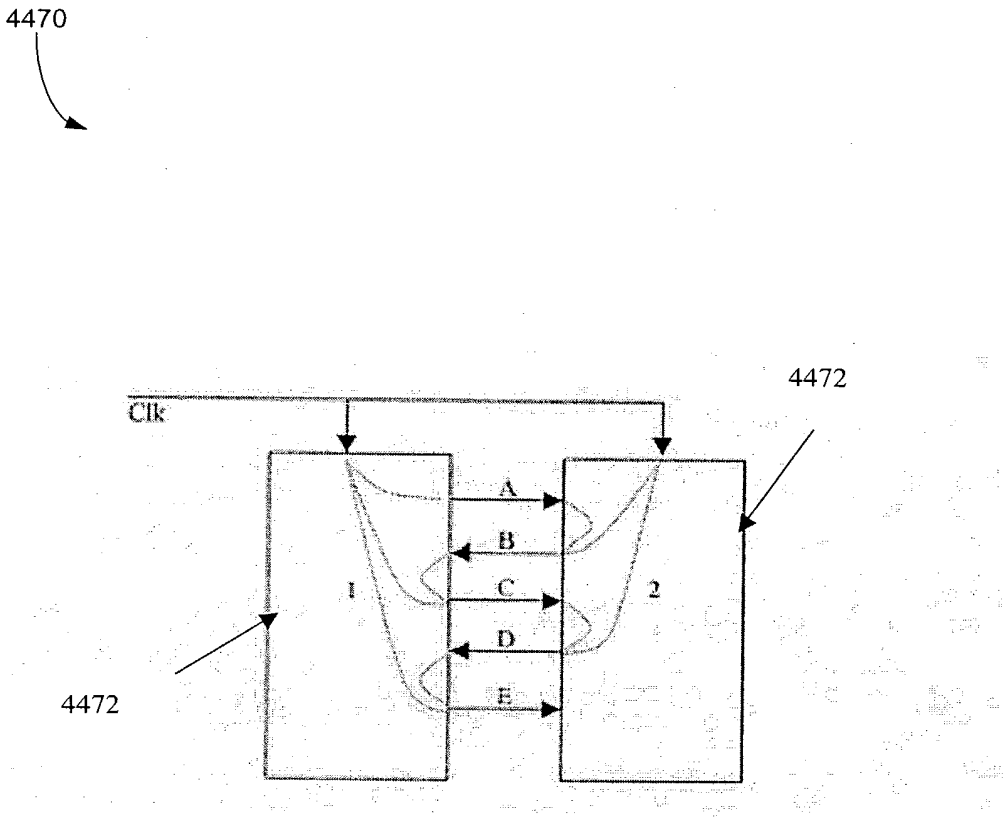



Fig. 44C

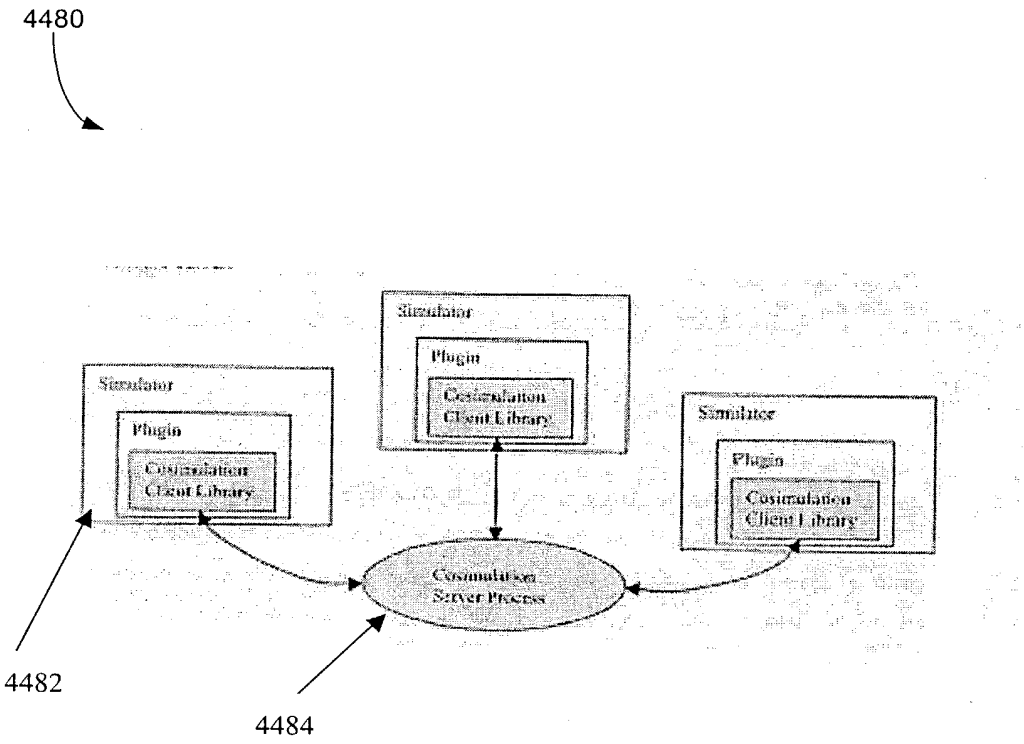

4480
Simulator
Plugin
Cosimulation
Client Library
Simulator
Plugin
Cosimulation
Client Library
Simulator
Plugin
Cosimulation
Client Library
Cosimulation
Server Process
4482
4484

Fig. 44D

4500

| Option | Meaning |
|---|---|
| -c | Compile only. Do not generate netlist. Output .obj file |
| -s | Target simulator |
| -cl*CommandLine* | Specify command line for compiling simulator output |
| -f*Family* | Specify target family |
| -p*Part* | Specify target part |
| -edif | Target EDIF output |
| -vhdl | Target VHDL output |
| -lpm *Width* | Use LPMs for data paths wider than *Width* |
| | |
| -b | Generate browse info database file |
| -r*Filename* | Specify browse info database file name |
| -o*Filename* | Specify output file name |
| -xc*Filename* | Treat file as Handel-C source file |
| -xl*Filename* | Treat file as Handel-C library file |
| -xo*Filename* | Treat file as Handel-C object file |
| -L*Pathname* | Add *pathname* to library path |
| | |
| -cpp *Option* | Pass *Option* to preprocessor |
| -D*Symbol* | Define preprocessor symbol |
| -E | Preprocess source only |
| -I*Pathname* | Add *pathname* to preprocessor include path |
| -U*Symbol* | Undefine preprocessor symbol |
| | |
| -O | Turn on maximum optimisations |
| -O- | Turn off all optimisations |
| -O+*optimise* | Turn on *optimise* optimisation |
| -O-*optimise* | Turn off *optimise* optimisation |
| | **Possible values for** *optimise* |
| | cr Conditional rewriting optimisations |

FIG. 45A

4500

| | | |
|---|---|---|
| | `cse` | CSE optimisations |
| | `high` | High-level optimisations. |
| | `pcse` | Partitioning CSE optimisations |
| | `rcse` | Repeated CSE optimisations. |
| | `rcr` | Repeated conditional rewriting optimisations. |
| | `retime` | Retiming optimisations. |
| | `rewrite` | Rewriting optimisations. |
| `-e` | Estimate logic depth and area. (Generate HTML files) | |
| `-g` | Compile with debug information | |
| `-W` | Show all warning messages | |
| `-help` | Print help screen | |

FIG. 45B

4600

| Command | Shortcut | Function |
|---|---|---|
| Edit | | |
| **New...** | Ctrl+N | Display the **New** dialog |
| **Open.....** | Ctrl+O | Display the **File Open** dialog |
| **Save..** | Ctrl+S | Save the active document |
| **Print** | Ctrl+P | Print active document |
| **Undo** | Ctrl+Z | Reverse the most recent change to the active document or to the workspace |
| **Redo** | Ctrl+Y | Reverse the most recent undo |
| **Cut** | Ctrl+X | Copy the current selection and delete it |
| **Copy** | Ctrl+C | Copy the current selection to the clipboard |
| **Paste** | Ctrl+V | Copy the clipboard to the current selection |
| **Delete** | Del | Delete the current selection |
| **Breakpoints...** | Alt+F9 | Display project's breakpoints dialog box |
| View | | |
| **Workspace** | Alt+0 | Hide or show the **Workspace** window |
| **Output** | Alt+2 | Hide or show the **Output** window |
| **Debug Windows** | | |
| **Watch** | Alt+3 | Hide or show the **Watch** window |
| **Call Stack** | Alt+7 | Hide or show the **Call Stack** window |
| **Memory** | Alt+6 | Hide or show the **Memory** window |
| **Variables** | Alt+4 | Hide or show the **Variables** window |
| **Clocks** | Alt+9 | Hide or show the **Clocks** window |
| **Threads** | Alt+5 | Hide or show the **Threads** window |
| **Properties** | Alt+Enter | Display properties of the current document or selection |
| Project | | |
| **Settings...** | Alt+F7 | Shows the **Project Settings** dialog box. |
| Build | | |
| **Compile** | Ctrl+F7 | Compiler selected file. |
| **Build** | F7 | Build this project |

FIG. 46A

4600

| Debug | | |
|---|---|---|
| **Go** | F5 | Run the simulator at full speed (until a breakpoint etc.) |
| **Restart** | Ctrl+Shift+F5 | Run the simulator from the beginning. |
| **Stop Debugging** | Shift+F5 | Stop the simulation |
| **Show Next Statement** | Alt + Num * | Run until the line containing the text cursor is reached. |
| **Step Into** | F11 | Run to the first statement in the function invoked in the current line. If the current line is not a function invocation, just run until the next statement. |
| **Step Over** | F10 | Run until the start of the next statement |
| **Step Out** | Shift+F11 | Run until the start of the statement after the line which invoked the current function |
| **Run to Cursor** | Ctrl+F10 | Run until the line containing the text cursor is reached. |
| **Advance** | Ctrl+F11 | Run until the line containing the text cursor is reached. |
| Tools | | |
| **Source Browser** | Alt+F12 | Show a symbol browser dialogue box. |
| Help | | |
| **Help Topics** | F1 | List the Help topics |
| | | |
| Output Window | | |
| | Double click | Takes you to line in source code |
| | F4 | Next error |
| | Shift + F4 | Previous error |
| Windows control | | |
| | F6 | Next pane |
| | Shift +F6 | Previous pane |

FIG. 46B

4700

**Directory browsing**

| | | | |
|---|---|---|---|
| | Folder | | Folder (open) |
| | Network disc | | Fixed disc |
| | Removable disc | | CDROM |
| | RAM disc | | |

**Output window**

| | | | |
|---|---|---|---|
| | Information | | Warning (about your program) |
| | User `assert` statement | | |
| | Error (in your program) | | Internal error in the compiler |
| | Position stack | | Position |

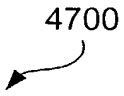
**Source window**

| | |
|---|---|
| | Current active point |
| | Other statements executed in current thread on current clock cycle |
| | Active point in different thread |
| | Position of current error |
| | Breakpoint |
| | Disabled breakpoint |

**Toolbar**

| | | | |
|---|---|---|---|
| | New Text File | | |
| | New ... | | Show Workspace window |
| | Open | | Show Output window |
| | Save | | Show properties |
| | Save All | | Compile one file |
| | Print | | Build project |
| | Cut | | Stop a build in progress |
| | Copy | | Run program in simulator |
| | Paste | | Add or remove breakpoint |
| | Undo | | Show about box |
| | Redo | | |

FIG. 47A




**Debug**

| | | | |
|---|---|---|---|
| | Restart, | | |
| | Stop debugging | | Pause |
| | Step in | | Step over, |
| | Step out, | | Advance |
| | Run to cursor | | |
| | Show watch windows | | Show variable windows |
| | Show call stack window | | |
| | Clock in clocks window | | |
| | Thread in threads window | | |

**Workspace window**

**File view**

- Handel-C workspace (`.hw` file)
- Handel-C system project (`.hp` file)
- Handel-C board project (`.hp` file)
- Handel-C chip project (`.hp` file)
- Handel-C core project (`.hp` file)
- Library project (`.hp` file)
- Document (`.txt`, `.h`)
- Source file (`.c` file)
- Folder
- Folder (open)

FIG. 47B

4700

**Symbol view**

- Target (used for configurations)
- Function or shared proc or expr
- In-line function, macro
- Variable
- RAM, ROM or WOM variable
- Channel (**chan**, **chan!** or **chan?**)
- Signal
- Interface
- Position stack
- Position

**Browsing**

- Go to definition
- Go to reference
- Previous
- Next
- Return to original context
- File outline
- Definitions and references

FIG. 47C

4800

| Raw file bit number | Colour bit |
|---|---|
| 7 (Most significant) | Red 7 |
| 6 | Green 7 |
| 5 | Blue 7 |
| 4 | Blue 6 |
| 3 | Green 6 |
| 2 | Red 6 |
| 1 | Green 5 |
| 0 (Least significant) | Green 4 |

FIG. 48

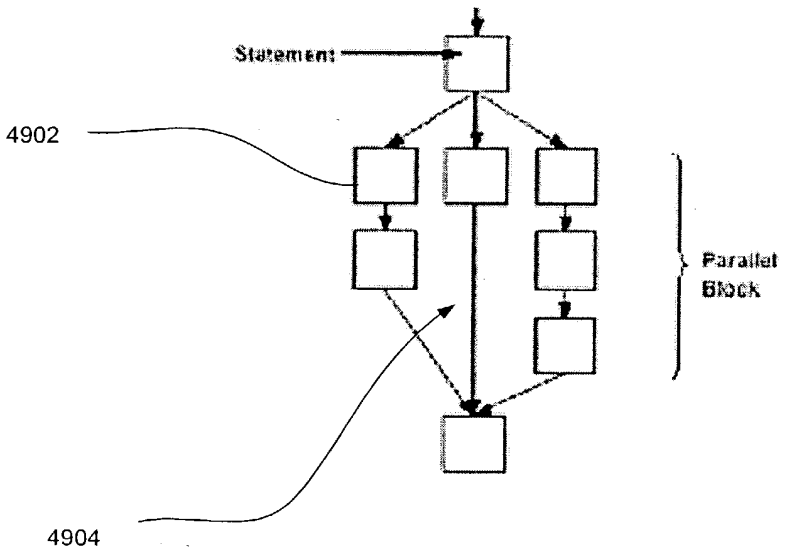



FIG. 49

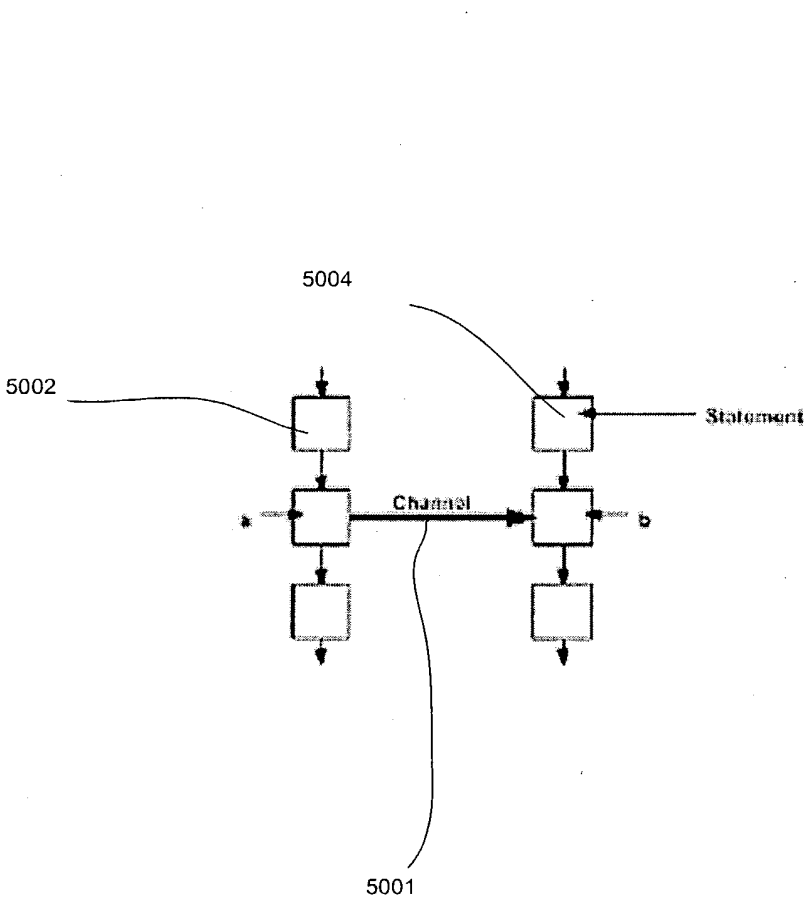



FIG. 50

5100

```
int w;

void main(void)
{
    int x;

    {
        int y;
        ------
    }
    {
        int z;
        ------
    }
}
```

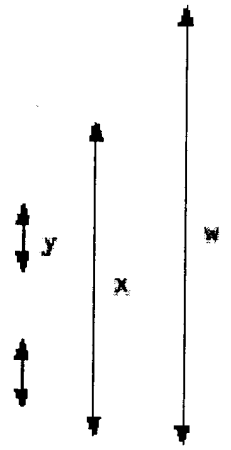



FIG. 51

5200

| Operator | Meaning | ISO-C | Change in Version 3 |
|---|---|---|---|
| [ ] | array index delimiters, bit selection | Extended | Array index may be a variable, bit selection may not |
| . | structure, union and multi-port RAM member operator, interface port operator | Yes | `struct` variables have been added |
| * | indirection operator | Yes | New |
| & | address operator | Yes | New |
| / | division operator | Yes | Extended: division of variables |
| % | modulus operator | Yes | Extended: modulus of variables |
| << | left-shift operator | Yes | Extended: shift by variable amounts |
| >> | right shift operator | Yes | Extended: shift by variable amounts |

FIG. 52

5300

**Declarations**

| Keyword | Meaning | ISO-C | Change in v.3 |
|---|---|---|---|
| **<>** | **disambiguator** | **No** | **New** |
| auto | auto variable | Yes | New |
| const | specify that variable's value will not change | Yes | New |
| enum | enumeration constant | Yes | New |
| extern | define global variable | Yes | New |
| **inline** | **declaration of inline function** | **No** | **New** |
| **interface** | **declaration of off-chip interface** | **No** | **Extended: you can now create interfaces to foreign code** |
| **mpram** | **declare a multi-port RAM** | **No** | **Create dual-ported RAMs** |
| **ram** | **declare a RAM** | **No** | **Extended to specify Xilinx block memory** |
| register | declare register variable | Yes | New |
| **rom** | **declare a ROM** | **No** | **Extended to specify Xilinx block memory** |
| **signal** | **declare a signal object** | **No** | **New** |
| signed | declare a signed variable | Yes | New |
| static | specify variable with limited scope | Yes | New |
| struct | declare a structure variable | Yes | New |
| typedef | define type | Yes | New |
| void | specify void return type or empty parameter list | Yes | New |
| volatile | declare volatile variable | Yes | New |
| **wom** | **declare a WOM (array)** | **No** | **Specify an area of write-only memory** |

FIG. 53

5400

**Statements**

| Statement | Meaning | ISO-C | Change in v.3 |
|---|---|---|---|
| `assert` | diagnostic macro to print to stderr | Not standard | Print string to standard error channel |
| `continue` | continue execution outside code block | Yes | New |
| `goto` | jump to specified label | Yes | New |
| `ifselect` | **conditional execution on compile time selection** | **No** | **Compile following code if selected, else...** |
| `par` | **execute statements in parallel** | **No** | **Extended: parallel statements can be replicated** |
| `return` | return from function | Yes | New |
| `seq` | **execute statements in sequence** | **No** | **New: seq blocks can also be replicated** |
| `typeof` | **return type of operator** | **No** | **As in GNU C** |

**Macros**

| Keyword | Meaning | ISO-C | Change in version 3 |
|---|---|---|---|
| `in` | **define scope for local macro expression declaration** | **No** | **New:** `let macro expr` *name* = *expression* `in` *macro expression* |
| `let` | **start declaration of local macro expression** | **No** | **New:** `let macro expr` *name* = *expression* `in` *macro expression* |

**Clocks**

| Keyword | Meaning | ISO-C | Change in version 3 |
|---|---|---|---|
| `internal` | **use internal clock** | **No** | **Extended: can use any expression** |
| `Internal_divide` | **use divided internal clock** | **No** | **Extended: can use any expression** |
| `__clock` | **use current clock** | **No** | **New** |

FIG. 54

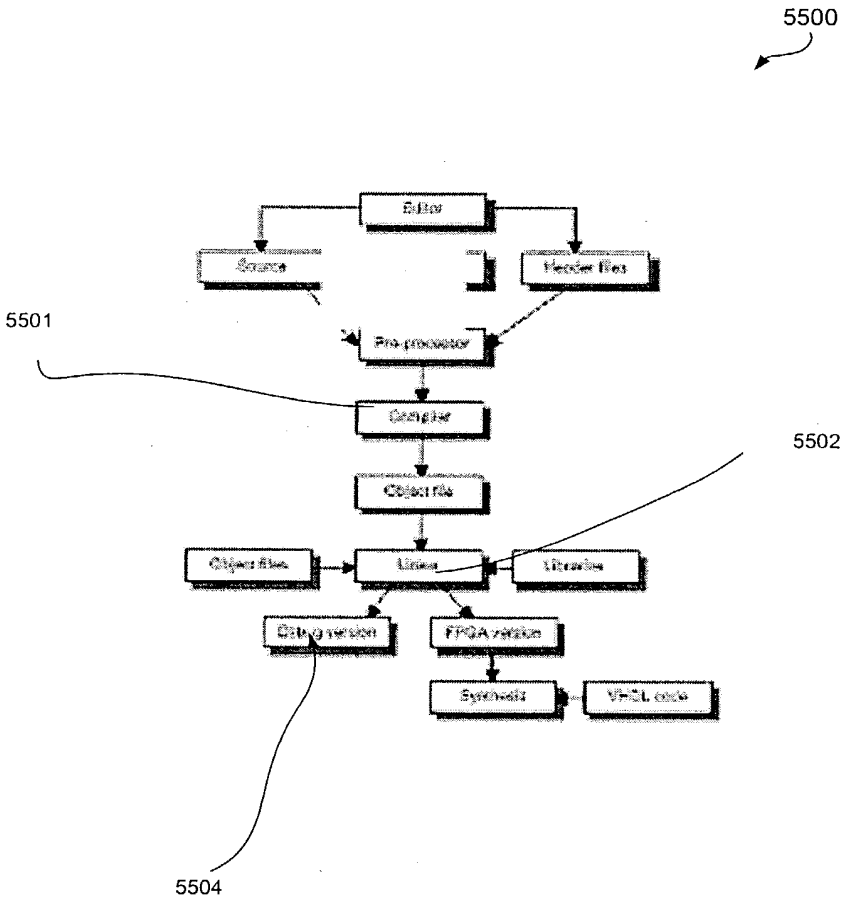



FIG. 55

5600

| Predefined bus interface specs: | | Default |
|---|---|---|
| data | list the pins used for transferring data, MSB to LSB | None |
| speed | set buffer speed (output) | Xilinx = 3<br>Altera = 1 |
| pull | set pull-up or pull-down for bus pins (not Altera) | None |
| infile | set file source for input bus data | None |
| outfile | set file destination for output bus data | None |

| All interface specs | | Default |
|---|---|---|
| base | specify display base for variables in debugger | 10 |
| extlib | specify external plugin for simulator | None |
| extfunc | specify external simulator function for this port | PlugInSet or PlugInGet |
| extpath | specify any direct logic (combinatorial logic) connections to another port | None |
| extinst | specify connection to external code | None |
| warn | disable some compiler warnings | No |

FIG. 56

5700

| ROM entry | Value | ROM entry | Value |
|---|---|---|---|
| b[0][0][0] | 1 | b[0][0][1] | 2 |
| b[0][1][0] | 3 | b[0][1][1] | 4 |
| b[1][0][0] | 5 | b[1][0][1] | 6 |
| b[1][1][0] | 7 | b[1][1][1] | 8 |
| b[2][0][0] | 9 | b[2][0][1] | 10 |
| b[2][1][0] | 11 | b[2][1][1] | 12 |
| b[3][0][0] | 13 | b[3][0][1] | 14 |
| b[3][1][0] | 15 | b[3][1][1] | 16 |

FIG. 57

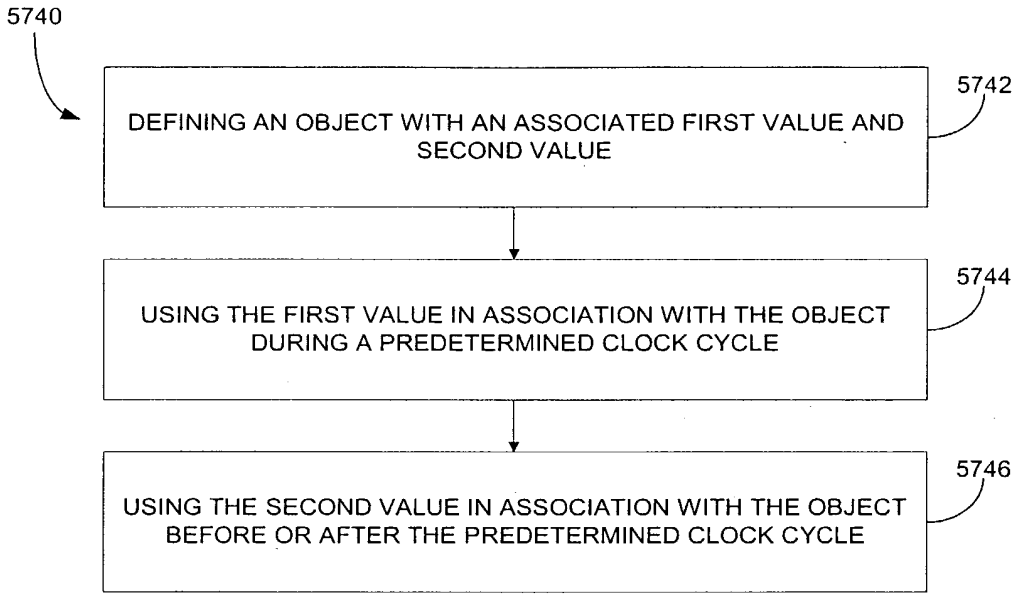

5740
DEFINING AN OBJECT WITH AN ASSOCIATED FIRST VALUE AND SECOND VALUE
5742
USING THE FIRST VALUE IN ASSOCIATION WITH THE OBJECT DURING A PREDETERMINED CLOCK CYCLE
5744
USING THE SECOND VALUE IN ASSOCIATION WITH THE OBJECT BEFORE OR AFTER THE PREDETERMINED CLOCK CYCLE
5746

Fig. 57A

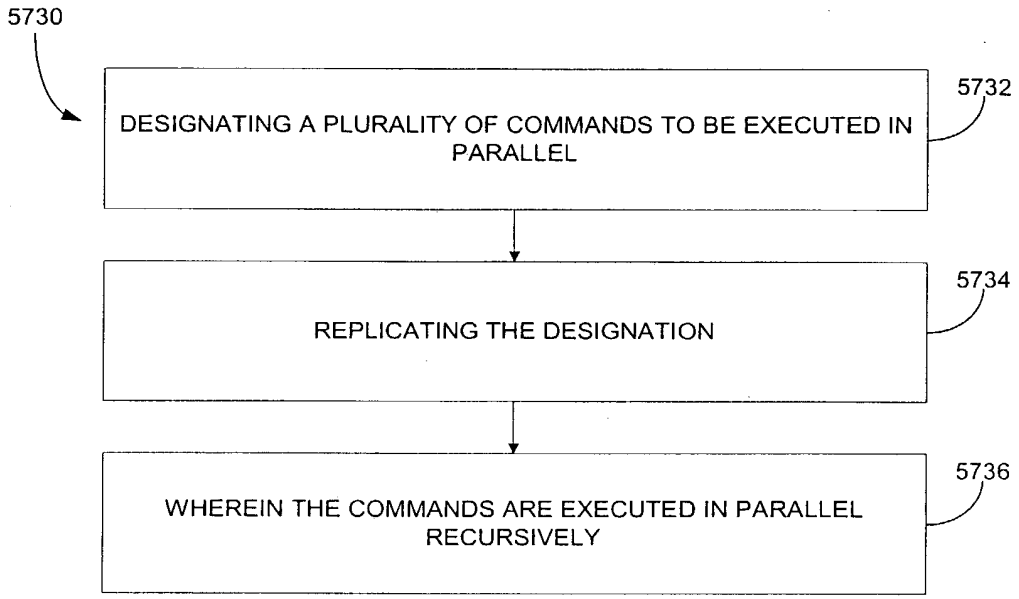

5730
DESIGNATING A PLURALITY OF COMMANDS TO BE EXECUTED IN PARALLEL
5732
REPLICATING THE DESIGNATION
5734
WHEREIN THE COMMANDS ARE EXECUTED IN PARALLEL RECURSIVELY
5736

Fig. 57A-1

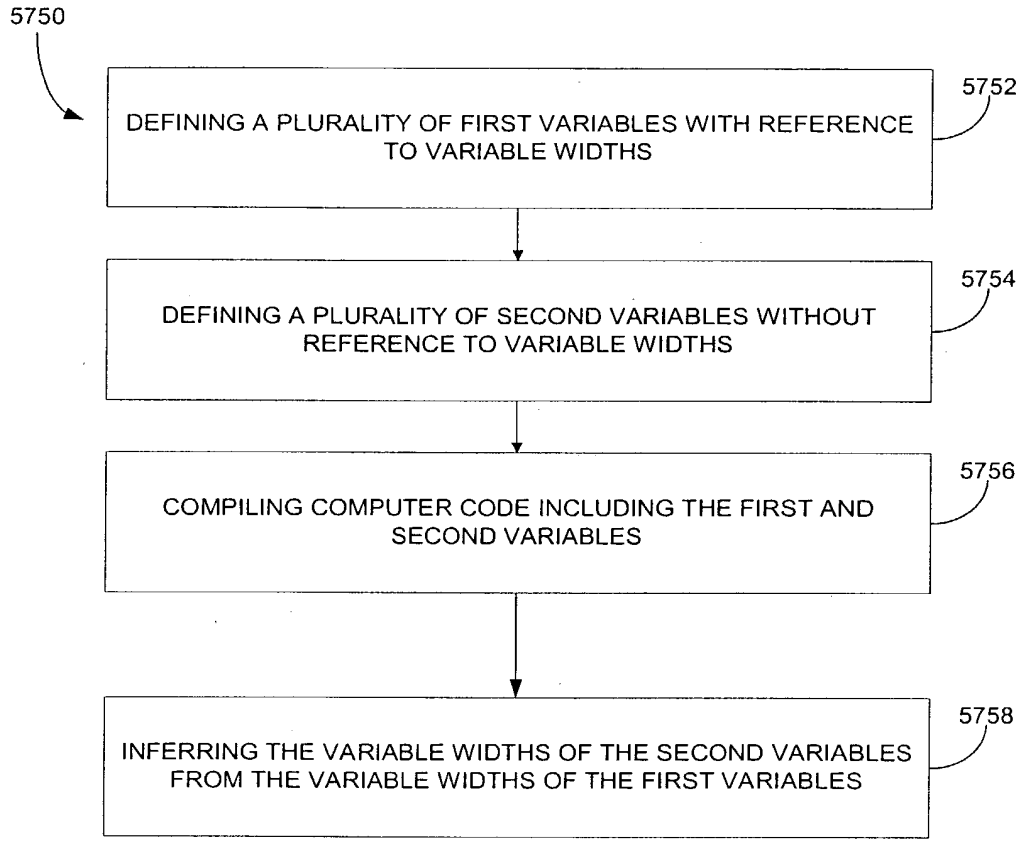

5750
DEFINING A PLURALITY OF FIRST VARIABLES WITH REFERENCE TO VARIABLE WIDTHS
5752
DEFINING A PLURALITY OF SECOND VARIABLES WITHOUT REFERENCE TO VARIABLE WIDTHS
5754
COMPILING COMPUTER CODE INCLUDING THE FIRST AND SECOND VARIABLES
5756
INFERRING THE VARIABLE WIDTHS OF THE SECOND VARIABLES FROM THE VARIABLE WIDTHS OF THE FIRST VARIABLES
5758

Fig. 57A-2

5800

| Statement | Timing |
|---|---|
| {...} | Sum of all statements in sequential block |
| `par` {...} | Length of longest branch in block |
| *Function*(), `break`, `goto`, `continue` | No clock cycles |
| `return`(*Expression*); | 1 clock cycle if *Expression* is assigned on return, otherwise none. |
| *Variable* = *Expression*; | 1 clock cycle |
| *Variable* ++; | 1 clock cycle |
| *Variable* --; | 1 clock cycle |
| ++ *Variable*; | 1 clock cycle |
| -- *Variable*; | 1 clock cycle |
| *Variable* += *Expression*; | 1 clock cycle |
| *Variable* -= *Expression*; | 1 clock cycle |
| *Variable* *= *Expression*; | 1 clock cycle |
| *Variable* /= *Expression*; | 1 clock cycle |
| *Variable* %= *Expression*; | 1 clock cycle |
| *Variable* <<= Constant; | 1 clock cycle |
| *Variable* >>= Constant; | 1 clock cycle |
| *Variable* &= *Expression*; | 1 clock cycle |
| *Variable* \|= *Expression*; | 1 clock cycle |
| *Variable* ^= *Expression*; | 1 clock cycle |
| *Channel* ? *Variable*; | 1 clock cycle when transmitter is ready (in same clock domain) |

FIG. 58A

5800

| Statement | Timing |
|---|---|
| *Channel* ! *Expression*; | 1 clock cycle when receiver is ready (in same clock domain) |
| `if (`*Expression*`) {...} else {...}` | Length of executed branch |
| `while (`*Expression*`) {...}` | Length of loop body * number of iterations |
| `do {...} while (`*Expression*`);` | Length of loop body * number of iterations |
| `for (`*Init*; *Test*; *Iter*`) {...}` | Length of *Init* + (Length of body + length of *Iter*) * number of iterations |
| `switch (`*Expression*`) {...}` | Length of executed case branch |
| `prialt {...}` | 1 clock cycle for case communication when other party is ready plus length of executed case branch<br>*or* length of default branch if present and no communication case is ready<br>*or* infinite if no default branch and no communication case is ready |
| `delay;` | 1 clock cycle |

FIG. 58B

5900

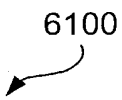
| Clock | in | x[0] | x[1] | x[2] | out |
|---|---|---|---|---|---|
| 1 | 5 | 0 | 0 | 0 | 0 |
| 2 | 6 | 5 | 0 | 0 | 0 |
| 3 | 7 | 6 | 5 | 0 | 0 |
| 4 | 8 | 7 | 6 | 5 | 0 |
| 5 | 9 | 8 | 7 | 6 | 5 |
| 5 | 10 | 9 | 8 | 7 | 6 |
| 6 | 11 | 10 | 9 | 8 | 7 |
| 7 | 12 | 11 | 10 | 9 | 8 |
| 8 | 13 | 12 | 11 | 10 | 9 |

FIG. 59

6000

| *Location* | **Meaning** |
|---|---|
| `internal` *Frequency*<br>`internal` *Expression* | Clock from internal clock generator (Xilinx 4000 series devices only). |
| `internal_divide` *Frequency Factor*<br>`internal_divide` *Expression* | Clock from internal clock generator with integer division (Xilinx 4000 series devices only). |
| `external` [*Pin*] | Clock from device pin. |
| `external_divide` [*Pin*] *Factor* | Clock from device pin with integer division. |

FIG. 60




| Family Name | Description |
|---|---|
| Xilinx4000E | 4000E series Xilinx FPGAs |
| Xilinx4000L | 4000L series Xilinx FPGAs |
| Xilinx4000EX | 4000EX series Xilinx FPGAs |
| Xilinx4000XL | 4000XL series Xilinx FPGAs |
| Xilinx4000XV | 4000XV series Xilinx FPGAs |
| XilinxVirtex | Virtex Xilinx FPGAs |
| Altera10K | Flex10K series Altera FPGAs |
| Altera20K | Flex20K series Altera FPGAs |

FIG. 61

6200

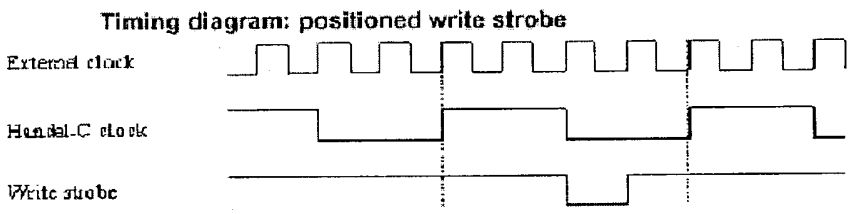

Timing diagram: positioned write strobe
External clock
Handel-C clock
Write strobe

FIG. 62

6300

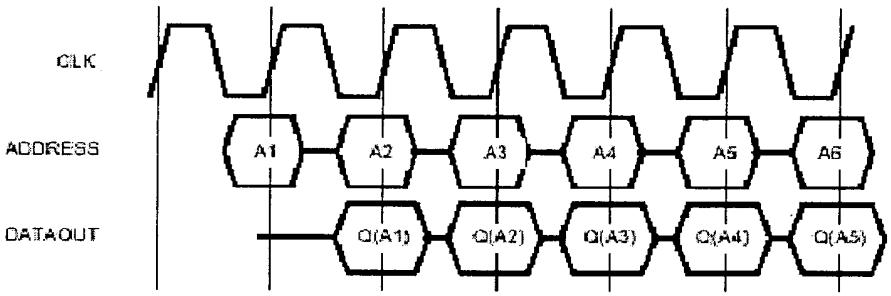



FIG. 63

6400

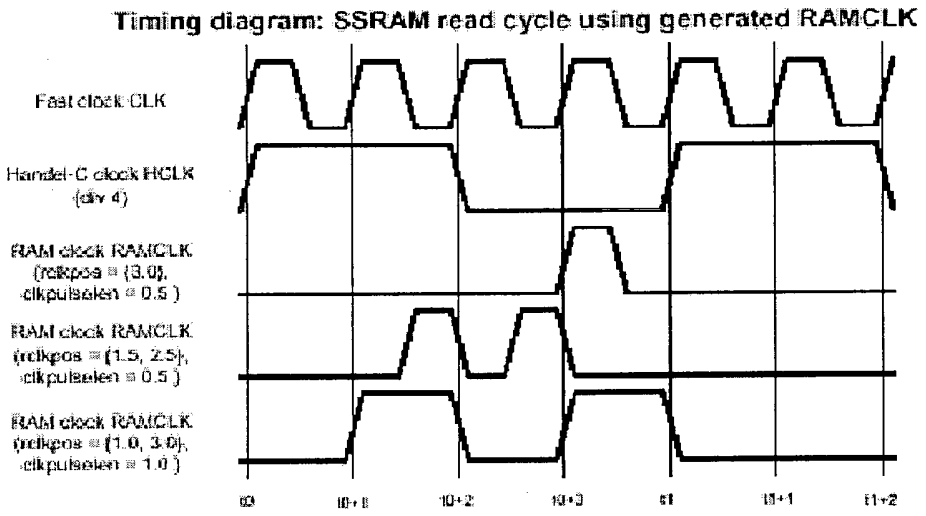

Timing diagram: SSRAM read cycle using generated RAMCLK
Fast clock CLK
Handel-C clock HCLK
(div 4)
RAM clock RAMCLK
(rclkpos = (3,0),
clkpulselen = 0.5 )
RAM clock RAMCLK
(rclkpos = (1.5, 2.5),
clkpulselen = 0.5 )
RAM clock RAMCLK
(rclkpos = (1.0, 3.0),
clkpulselen = 1.0 )

FIG. 64

6500

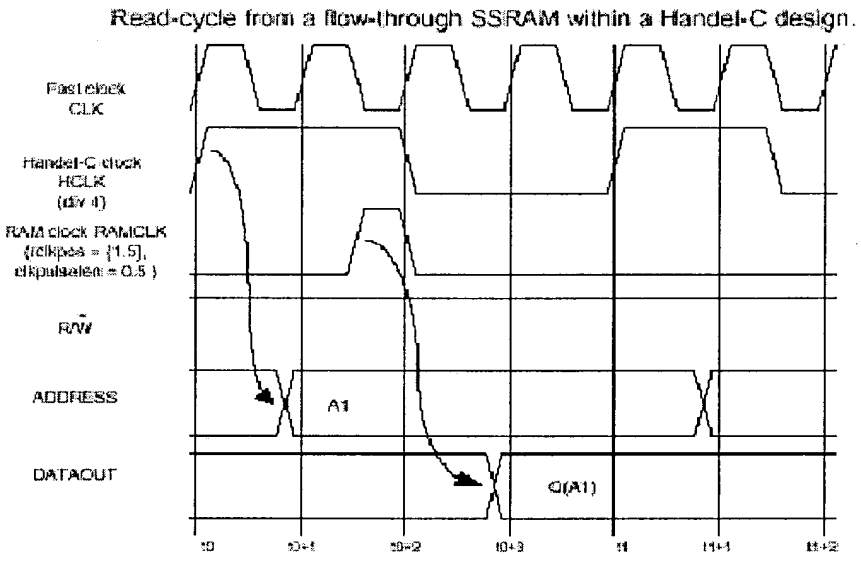

Read-cycle from a flow-through SSRAM within a Handel-C design.
Fast clock CLK
Handel-C clock HCLK (div 4)
RAM clock RAMCLK (rclkpos = {1.5}, clkpulselen = 0.5 )
R/W
ADDRESS
DATAOUT
A1
Q(A1)
t0
t0+1
t0+2
t0+3
t1
t1+1
t1+2

FIG. 65

6600

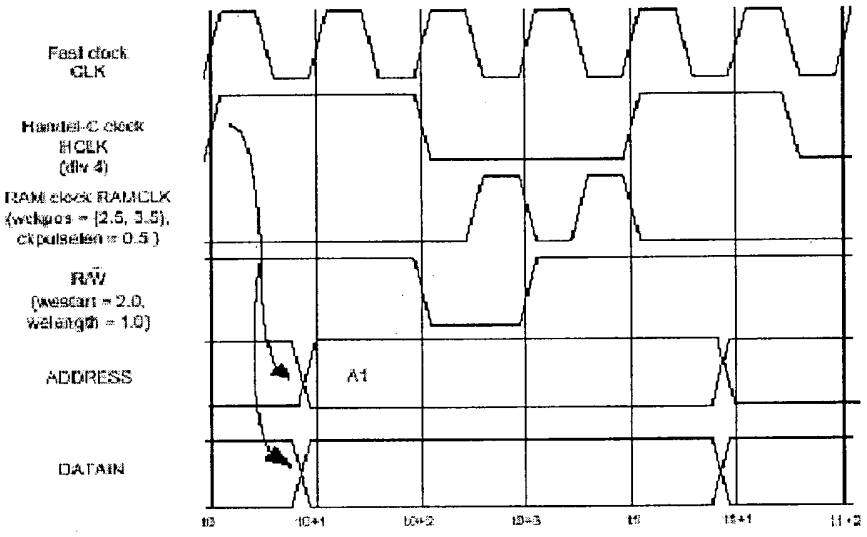

Fast clock
CLK
Handel-C clock
HCLK
(div 4)
RAM clock RAMCLK
(wclkpos = (2.5, 3.5),
ckpulselen = 0.5 )
R/W
(westart = 2.0,
welength = 1.0)
ADDRESS
A1
DATAIN

FIG. 66

6700

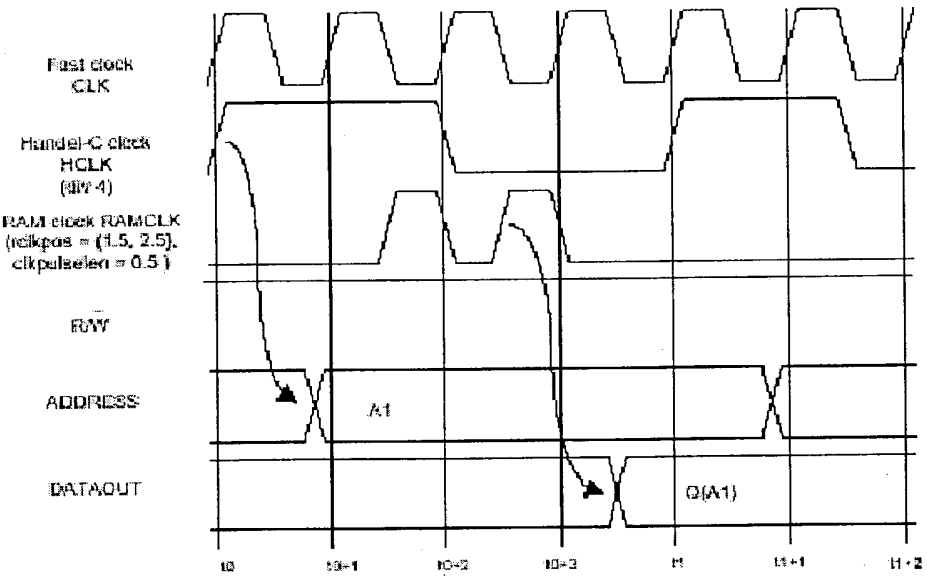

Fast clock
CLK
Handel-C clock
HCLK
(div 4)
RAM clock RAMCLK
(rclkpos = (1.5, 2.5),
clkpulselen = 0.5 )
R/W
ADDRESS
DATAOUT
A1
Q(A1)
t0
t0+1
t0+2
t0+3
t1
t1+1
t1+2

FIG. 67

6800

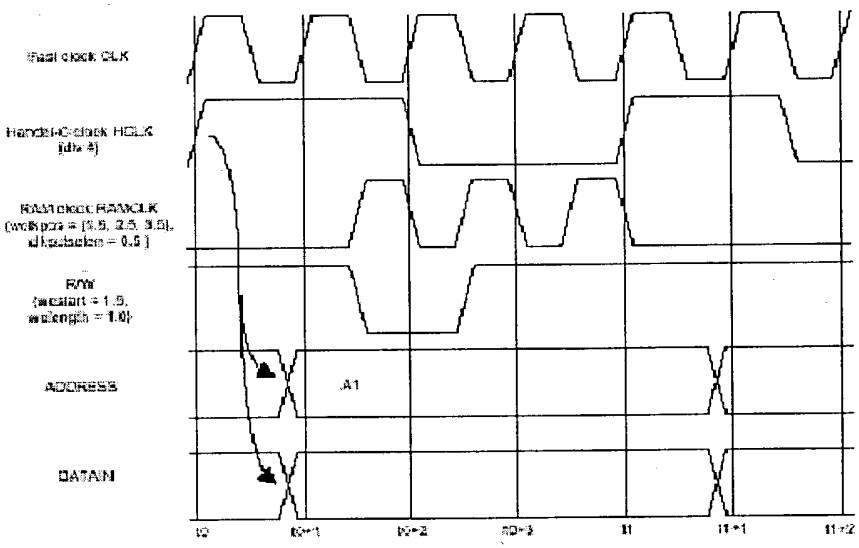

fast clock CLK
HandelC clock HCLK
(div 4)
RAM clock RAMCLK
(wstart = 0.5, 2.5, 3.5,
clkskew = 0.5 )
R/W
(wstart = 1.5,
wlength = 1.0)
ADDRESS
A1
DATAIN
I0
I0+1
I0+2
I0+3
I1
I1+1
I1+2

FIG. 68

6900

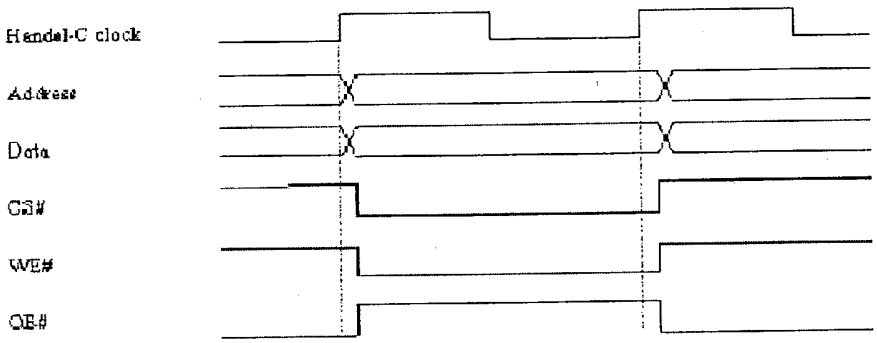



FIG. 69

7000

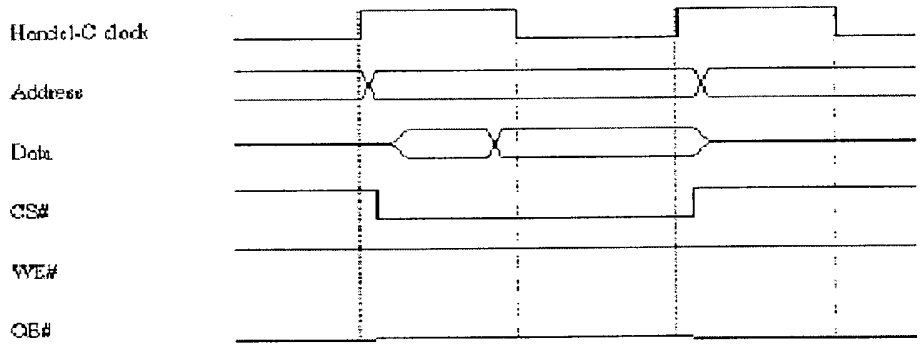



FIG. 70

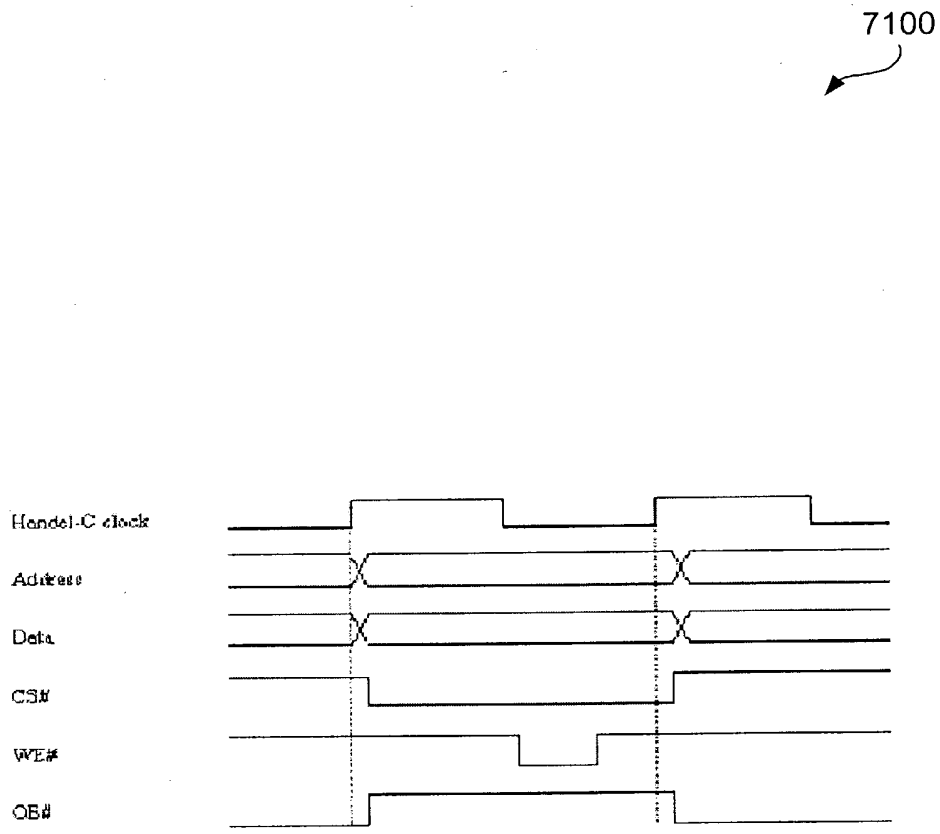



FIG. 71

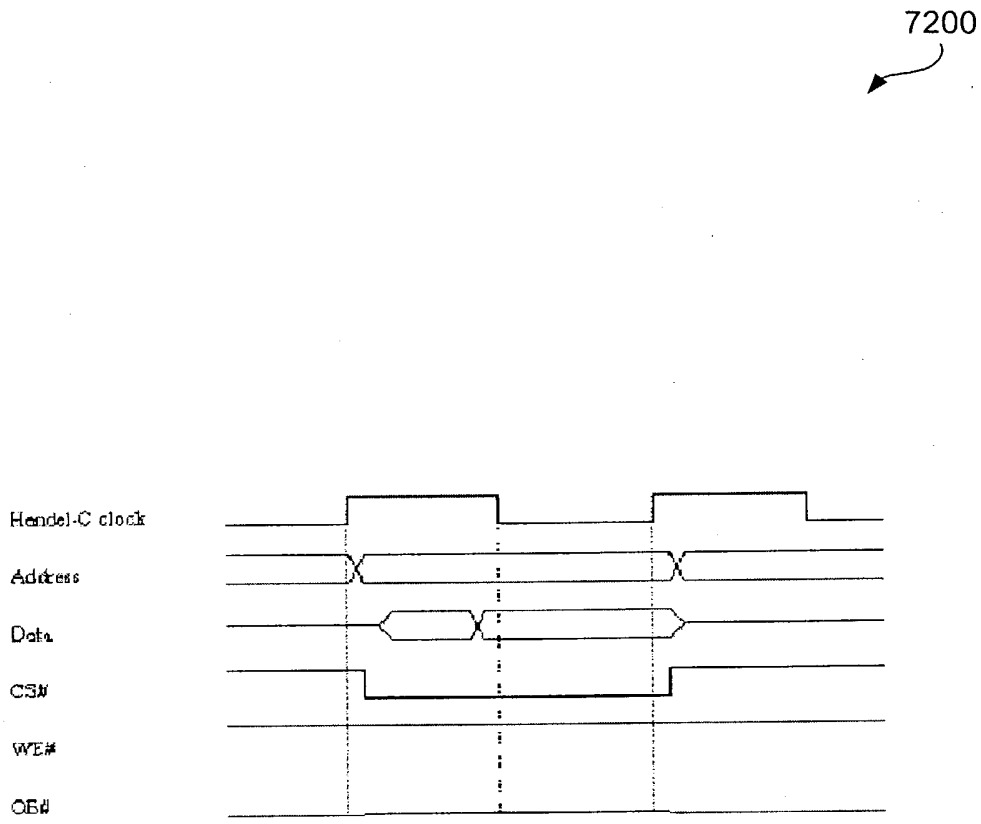



FIG. 72

7300

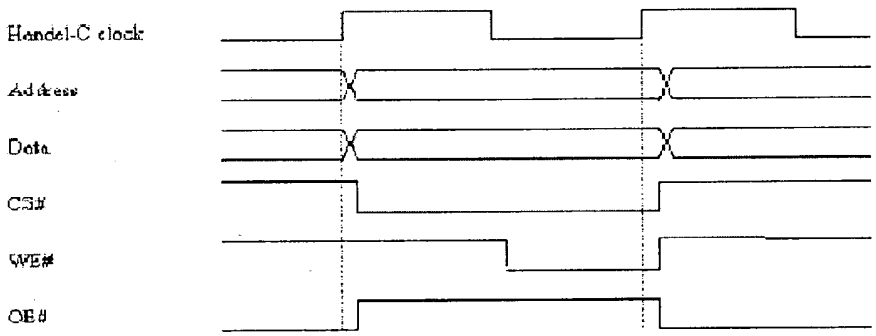



FIG. 73

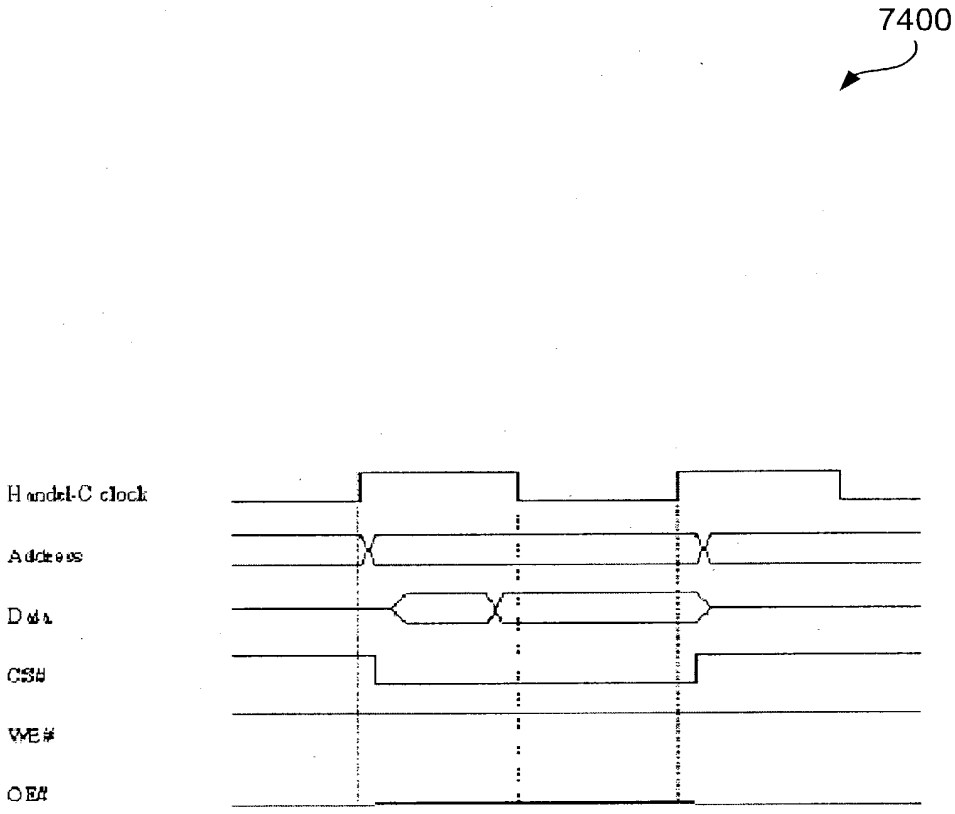



FIG. 74

7500

| Sort Identifier | Description |
|---|---|
| bus_in | Input bus from pins |
| bus_latch_in | Registered input bus from pins |
| bus_clock_in | Clocked input bus from pins |
| bus_out | Output bus to pins |
| bus_ts | Bi-directional tri-state bus |
| bus_ts_latch_in | Bi-directional tri-state bus with registered input |
| bus_ts_clock_in | Bi-directional tri-state bus with clocked input |
| port_in | Input port from logic |
| port_out | Output port to logic |

FIG. 75

7600

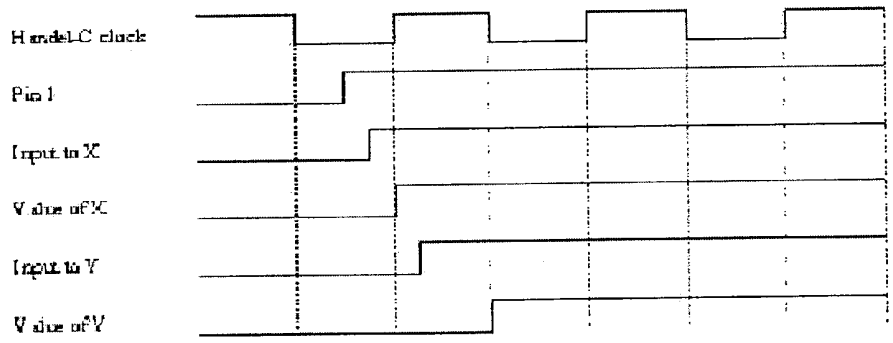



FIG. 76

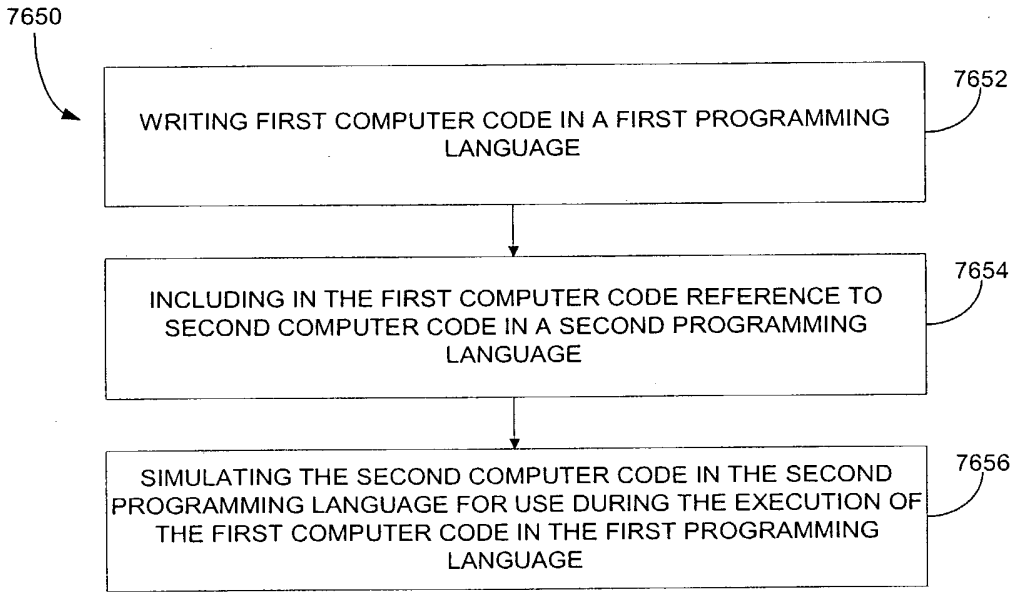

7650
WRITING FIRST COMPUTER CODE IN A FIRST PROGRAMMING LANGUAGE
7652
INCLUDING IN THE FIRST COMPUTER CODE REFERENCE TO SECOND COMPUTER CODE IN A SECOND PROGRAMMING LANGUAGE
7654
SIMULATING THE SECOND COMPUTER CODE IN THE SECOND PROGRAMMING LANGUAGE FOR USE DURING THE EXECUTION OF THE FIRST COMPUTER CODE IN THE FIRST PROGRAMMING LANGUAGE
7656

Fig. 76A

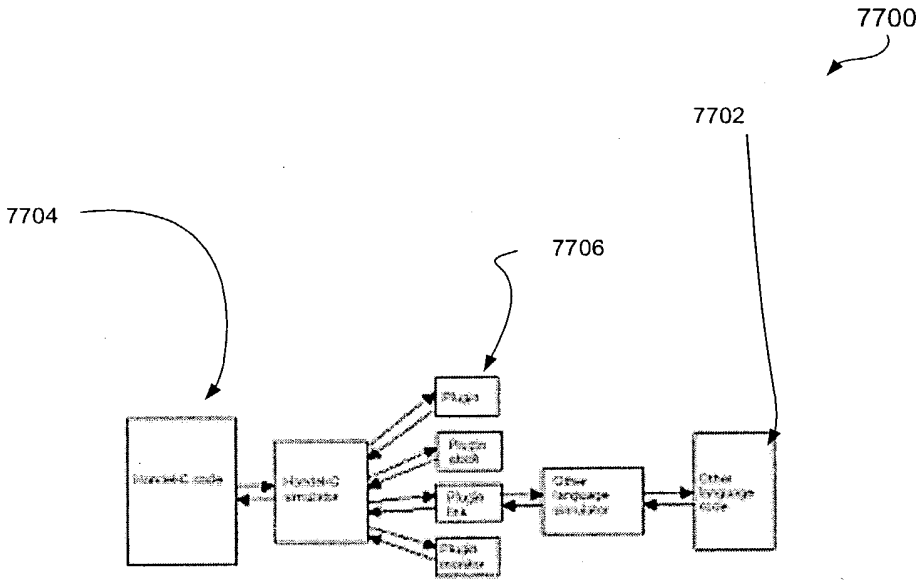



FIG. 77

7800

| Specification | Possible Values | Default | Applies to | Meaning |
|---|---|---|---|---|
| `show` | 0, 1 | 1 | variables<br>channels<br>o/p buses<br>tri-state buses | Show variable during simulation |
| `base` | 2, 8, 10, 16 | 10 | variables<br>o/p channels<br>o/p buses<br>tri-state buses | Print variable in specified base |
| `infile` | Any valid filename | None | chanins<br>i/p buses<br>tri-state buses | Redirect from file |
| `outfile` | Any valid filename | None | chanouts<br>o/p buses<br>tri-state buses | Redirect to file |
| `warn` | 0, 1 | 1 | variables<br>memories<br>channels<br>buses | Enable warnings for object |
| `speed` | 0, 1, 2, 3 | Xilinx = 3<br>Altera = 1 | o/p or tri-state bus | Set buffer speed |
| `intime` | Any floating point ns delay | None | input port or bus or tri-state bus<br>external RAMs | Maximum allowable delay between interface and variable |
| `outtime` | Any floating point ns delay | None | output port or bus or tri-state bus<br>external RAMs | Maximum allowable delay between variable and interface |
| `extlib` | Name of a plugin `.dll` | None | interface or port | Specify external plugin for simulator |

FIG. 78A

7800

| Specification | Possible Values | Default | Applies to | Meaning |
|---|---|---|---|---|
| extfunc | Name of a function within the plugin | PlugInSet or PlugInGet depending on port direction | interface or port | Specify external function within the simulator for this port |
| extpath | Name of port TO Handel-C on the same interface | None | port FROM Handel-C | Specify any direct logic (combinatorial logic) connections to another port |
| extinst | Instance name (with optional parameters) | None | interface or port | Specify simulation instance used |
| busformat | Format string | B_1 | interface, port or memories in external logic | Specify the way that wire names are formatted in EDIF |
| pull | 0, 1 | None | Xilinx buses | Add pull up or pull down resistor(s) |
| data | Any valid pin list | None | memories buses | Set data pins |
| offchip | 0, 1 | 0 | memories | Set RAM/ROM to be off chip |
| ports | 0, 1 | 0 | memories | Set RAM/ROM to be in external code |
| block | 0, 1 | Xilinx =0 Altera=1 | memories (on-chip) | Set RAM/ROM to be in block memory |
| wegate | -1, 0, 1 | 0 | RAMs | Place write enable signal |
| westart | 0 to clock division -1 | None | RAMs | Position write enable signal |
| welength | 1 to clock division | None | RAMs | Set length of write enable signal |
| rclkpos | Any number of cycles or half-cycles | None | SSRAM | Set read cycle position of SSRAM clock |
| wclkpos | Any number | None | SSRAM | Set write cycle |

FIG. 78B

7800

| Specification | Possible Values | Default | Applies to | Meaning |
|---|---|---|---|---|
| | of cycles or half-cycles | | | position of SSRAM clock |
| clkpulselen | Any number of cycles or half-cycles | None | SSRAM | Set pulse length of SSRAM clock |
| clk | Any valid pin list | None | SSRAM (off-chip) | Set clock pins for external SSRAM clock |
| addr | Any valid pin list | None | memories (off-chip) | Set address pins |
| oe | Any valid pin list | None | memories (off-chip) | Set output enable pin(s) |
| we | Any valid pin list | None | RAMs (off-chip) | Set write enable pin(s) |
| cs | Any valid pin list | None | memories (off-chip) | Set chip select pin(s) |
| rate | Any floating point ns delay | None | clock | Maximum allowable inter-component delay |

FIG. 78C

7850

| Specification | Input bus | Output bus | Tri-state bus | RAM | ROM |
|---|---|---|---|---|---|
| addr | | | | ✓ | ✓ |
| data | ✓ | ✓ | ✓ | ✓ | ✓ |
| we | | | | ✓ | |
| cs | | | | ✓ | ✓ |
| oe | | | | ✓ | ✓ |
| clk | | | | ✓ | |

FIG. 78D

7900

| Signal Name | FPGA pin | Description |
|---|---|---|
| D3..0 | 1, 2, 3, 4 | Data Bus |
| Write | 5 | Write strobe |
| Read | 6 | Read strobe |
| WriteRdy | 7 | Able to write to device |
| ReadRdy | 8 | Able to read from device |

FIG. 79

8000

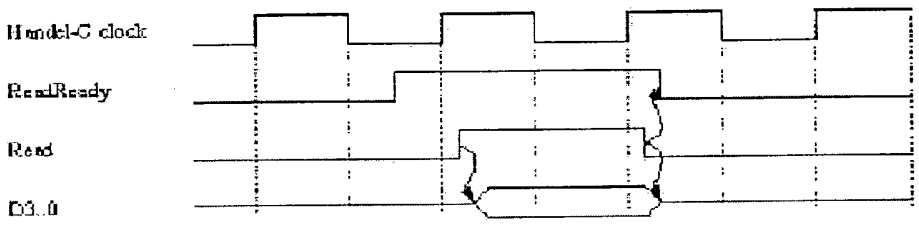



FIG. 80

8100

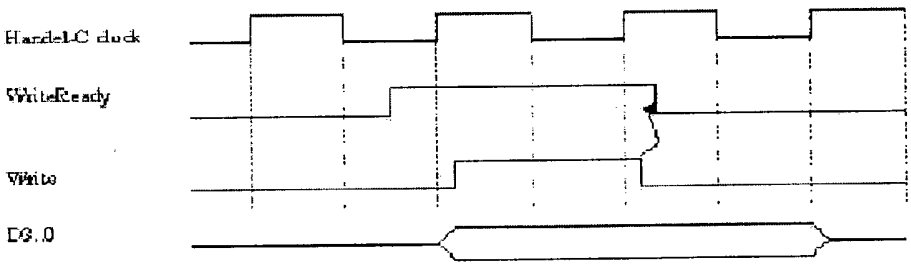



FIG. 81

8200

| Type | Width |
|---|---|
| `[signed \| unsigned] char` | 8 bits |
| `[signed \| unsigned] short` | 16 bits |
| `[signed \| unsigned] long` | 32 bits |
| `[signed \| unsigned] int` | See note 1 |
| `[signed \| unsigned] int` *n* | *n* bits |
| `[signed \| unsigned] int undefined` | Compiler infers width |
| `typeof` (*Expression*) | Yields type of object |

Note 1: Width will be inferred by compiler unless the '`set intwidth` = *n*' command appears before the declaration

FIG. 82

8300

| Prefix | Object |
|---|---|
| chan | Channel |
| chanin | Simulator channel |
| chanout | Simulator channel |
| ram | Internal or external RAM |
| rom | Internal or external ROM |
| signal | Wire |
| wom | WOM within multi-port memory |

The compound types are

| Prefix | Object |
|---|---|
| struct | Structure |
| mpram | Multi-port memory |

FIG. 83

8400

| Statement | Meaning |
|---|---|
| `par {...}` | Parallel composition |
| `seq {...}` | Sequential execution |
| `par (Init ; Test ; Iter){...}` | Parallel replication |
| `seq (Init ; Test ; Iter){...}` | Sequential replication |
| *Variable* = *Expression*; | Assignment |
| *Variable* ++; | Increment |
| *Variable* --; | Decrement |
| ++ *Variable*; | Increment |
| -- *Variable*; | Decrement |
| *Variable* += *Expression*; | Add and assign |
| *Variable* -= *Expression*; | Subtract and assign |
| *Variable* *= *Expression*; | Multiply and assign |
| *Variable* /= *Expression*; | Divide and assign |
| *Variable* %= *Expression*; | Modulus and assign |
| *Variable* <<= *Expression*; | Shift left and assign |
| *Variable* >>= *Expression*; | Shift right and assign |
| *Variable* &= *Expression*; | AND and assign |
| *Variable* \|= *Expression*; | OR and assign |
| *Variable* ^= *Expression*; | XOR and assign |
| *Channel* ? *Variable*; | Channel input |
| *Channel* ! *Expression*; | Channel output |
| `if (`*Expression*`) [{...} else {...}]` | Conditional execution |
| `ifselect (`*Expression*`) [{...} else {...}]` | Conditional compilation |
| `while (`*Expression*`) {...}` | Iteration |
| `do {...} while (`*Expression*`);` | Iteration |
| `for (`*Init* ; *Test* ; *Iter*`) {...}` | Iteration |
| `break;` | Loop and switch termination |
| `continue;` | Resume execution |
| `return[([`*Expression*`])];` | Return from function |
| `goto` *label*; | Jump to label |
| `switch (`*Expression*`) {...}` | Selection |
| `prialt {...}` | Channel alternation |
| `delay;` | Single cycle delay |

Note: RAM and ROM elements, signals and array elements are included in the set of variables in this table.

FIG. 84

8500

| Operator | Meaning |
|---|---|
| select(*Constant*, *Expr*, *Expr*) | Compile-time selection |
| *Expression* [*Expression*] | Array or memory subscripting |
| *Expression* [*Constant*] | Bit selection |
| *Expression* [*Constant*:*Constant*] | Bit range extraction. One of the two constants may be omitted |
| *functionName* (*Arguments*) | Function call[1] |
| *pointertostructure*->*member* | Structure reference |
| *structureName*.*member* | Structure reference |
| ! *Expression* | Logical NOT |
| ~ *Expression* | Bitwise NOT |
| - *Expression* | Unary minus |
| + *Expression* | Unary plus |
| & *object* | Yields pointer to operand |
| * *pointer* | Yields object or function that the operand points to |
| (*Type*) *Expression* | Type casting |
| width(*Expression*) | Width of expression |
| *Expression* <- *Constant* | Take LSBs |
| *Expression* \\ *Constant* | Drop LSBs |
| *Expression* * *Expression* | Multiplication |
| *Expression* / *Expression* | Division |
| *Expression* % *Expression* | Modulus arithmetic |
| *Expression* + *Expression* | Addition |
| *Expression* - *Expression* | Subtraction |
| *Expression* << *Expression* | Shift left |
| *Expression* >> *Expression* | Shift right |
| *Expression* @ *Expression* | Concatenation |
| *Expression* < *Expression* | Less than |
| *Expression* > *Expression* | Greater than |

FIG. 85A

8500

| Operator | Meaning |
|---|---|
| *Expression* <= *Expression* | Less than or equal |
| *Expression* >= *Expression* | Greater than or equal |
| *Expression* == *Expression* | Equal |
| *Expression* != *Expression* | Not equal |
| *Expression* & *Expression* | Bitwise AND |
| *Expression* ^ *Expression* | Bitwise XOR |
| *Expression* \| *Expression* | Bitwise OR |
| *Expression* && *Expression* | Logical AND |
| *Expression* \|\| *Expression* | Logical OR |
| *Expression* ? *Expr* : *Expr* | Conditional selection |

FIG. 85B

8600

| Keyword | Meaning | ISO-C | New in version 3 |
|---|---|---|---|
| = | assignment operator | Yes | |
| ; | statement terminator | Yes | |
| , | C only | Yes | |
| { } | code block delimiters | Yes | |
| **<>** | **type specialisation** | **No** | **New** |
| ( | open delimiter | Yes | |
| ) | close delimiter | Yes | |
| [ ] | array index delimiters, bit selection | Yes | Array index may be variable |
| **[ : ]** | **bit range selection** | **No** | **Extended** |
| ! | logical NOT operator | Yes | |
| **!** | **output to channel** | **No** | |
| ~ | bitwise NOT | Yes | |
| + | addition operator | Yes | |
| - | subtraction operator | Yes | |
| - | unary minus operator | Yes | |
| * | multiplication operator | Yes | |
| / | division operator | Yes | Extended |
| % | modulus operator | Yes | Extended |
| **** | **drop LSB** | **No** | |
| **<-** | **take LSBs** | **No** | |
| **?** | **read from channel** | **No** | |
| ? | conditional expression | Yes | |
| ^ | Bitwise XOR | Yes | |
| & | Bitwise AND | Yes | |
| \| | Bitwise OR | Yes | |

FIG. 86A

8600

| Keyword | Meaning | ISO-C | New in version 3 |
|---|---|---|---|
| && | Logical AND | Yes | |
| \|\| | Logical OR | Yes | |
| . | structure member operator | Yes | New |
| << | left-shift operator | Yes | Extended |
| >> | right shift operator | Yes | Extended |
| < | less than operator | Yes | |
| > | greater than operator | Yes | |
| <= | less or equal operator | Not standard[1] | |
| >= | greater or equal operator | Not standard[1] | |
| == | equality operator | Not standard[1] | |
| != | inequality operator | Not standard[1] | |
| ++ | increment operator | Not standard | |
| -- | decrement operator | Not standard | |
| += | assignment operator | Not standard | |
| -= | assignment operator | Not standard | |
| *= | assignment operator | Not standard | |
| /= | assignment operator | Not standard | |
| %= | assignment operator | Not standard | |
| <<= | assignment operator | Not standard | |
| >>= | assignment operator | Not standard | |
| &= | assignment operator | Not standard | |
| \|= | assignment operator | Not standard | |
| ^= | assignment operator | Not standard | |
| ... | Reserved. Not valid in Handel-C | C only | |
| -> | structure pointer operator | Yes | New |
| @ | **concatenation operator** | **No** | |

FIG. 86B

8600

| Keyword | Meaning | ISO-C | New in version 3 |
|---|---|---|---|
| assert | diagnostic macro to print to stderr | Not standard | New |
| auto | auto variable | Yes | New |
| break | immediate exit from code block | Yes | |
| case | selection within switch | Yes | |
| chan | **define channel variable** | **No** | |
| chanin | **simulator channel in** | **No** | |
| chanout | **simulator channel out** | **No** | |
| char | 8-bit variable | Yes | |
| clock | **define clock** | **No** | |
| const | specify that variable's value will not change | Yes | New |
| continue | force next iteration of loop | Yes | New |
| default | default case within switch, prialt | Yes | |
| delay | **wait one clock tick** | **No** | |
| do | start do while loop | Yes | |
| double | Reserved. Not valid in Handel-C | C-only | |
| else | conditional execution | Yes | |
| enum | enumeration constant | Yes | New |
| expr | **define macro as expression** | **No** | |
| extern | define global variable | Yes | New |
| external | **clock from device pin** | **No** | **Extended** |
| external_divide | **clock from device pin with integer division** | **No** | **Extended** |
| family | **define target device's family** | **No** | |
| float | Reserved. Not valid in Handel-C | C-only | |
| for | for loop iteration | Yes | |
| goto | jump to specified label | Yes | New |
| if | conditional execution | Yes | |
| ifselect | **conditional compilation on compile-time selection** | **No** | **New** |
| in | **define scope for local macro expression declaration** | **No** | **New** |
| inline | **declaration of inline function** | **No** | **New** |

FIG. 86C

8600

| Keyword | Meaning | ISO-C | New in version 3 |
|---|---|---|---|
| int | definable width variable | Yes | |
| interface | declaration of off-chip interface | No | Extended |
| internal | use internal clock | No | Extended |
| internal_divide | internal clock with integer division | No | Extended |
| intwidth | set integer width | No | |
| let | start declaration of local macro expression | No | New |
| long | declare 32-bit variable | Yes | |
| macro | declare a macro | No | |
| mpram | declare a multi-port RAM | No | New |
| par | execute statements in parallel | No | Extended |
| part | define target hardware | No | |
| prialt | execute first ready channel | No | |
| proc | define macro as procedure | No | |
| ram | declare a RAM (array) | No | |
| register | declare register variable | Yes | New |
| return | return from function | Yes | New |
| rom | declare a ROM (array) | No | |
| select | select expression or macro expr at compile time | No | |
| set | specify int width, target or clock | No | |
| seq | execute statements in sequence | No | New |
| shared | declare a shared expression | No | |
| short | declare 16-bit variable | Yes | |
| signal | declare a signal object | No | New |
| signed | declare a signed variable | Yes | New |
| sizeof | | | |
| sizeof | Reserved. Not valid in Handel-C | Yes | |
| static | specify variable with limited scope | Yes | New |
| struct | declare a structure variable | Yes | New |
| switch | switch statement (between cases) | Yes | |
| typedef | define type | Yes | New |
| typeof | return type of operator | No | New |

FIG. 86D

8600

| Keyword | Meaning | ISO-C | New in version 3 |
|---|---|---|---|
| undefined | specify a variable of undefined width | No | |
| union | | | |
| unsigned | declare an unsigned variable | Yes | |
| void | specify void return type, | Yes | New |
| volatile | declare volatile variable | Yes | New |
| while | loop statement | Yes | |
| width | return integer width | No | |
| with | specify interface, signals, channels, ram and rom types, variables etc. | No | |
| wom | declare a WOM (array) | No | New |

FIG. 86E

8700

| Escape Code | ASCII Value | Meaning |
|---|---|---|
| \a | 7 | Bell (alert) |
| \b | 8 | Backspace |
| \f | 12 | Form feed |
| \t | 9 | Horizontal tab |
| \n | 10 | Newline |
| \v | 11 | Vertical tab |
| \r | 13 | Carriage return |
| \" | - | Double quote mark |
| \0 | 0 | String terminator |
| \\ | - | Backslash |
| \' | - | Single quote mark |
| \? | - | Question mark |

FIG. 87A

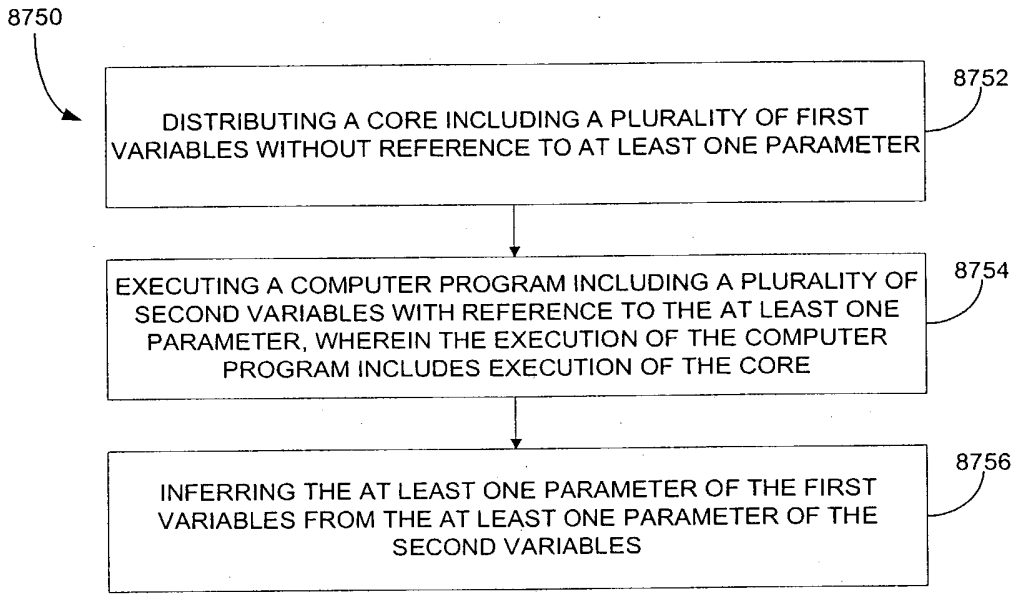

8750
DISTRIBUTING A CORE INCLUDING A PLURALITY OF FIRST VARIABLES WITHOUT REFERENCE TO AT LEAST ONE PARAMETER
8752
EXECUTING A COMPUTER PROGRAM INCLUDING A PLURALITY OF SECOND VARIABLES WITH REFERENCE TO THE AT LEAST ONE PARAMETER, WHEREIN THE EXECUTION OF THE COMPUTER PROGRAM INCLUDES EXECUTION OF THE CORE
8754
INFERRING THE AT LEAST ONE PARAMETER OF THE FIRST VARIABLES FROM THE AT LEAST ONE PARAMETER OF THE SECOND VARIABLES
8756

Fig. 87B

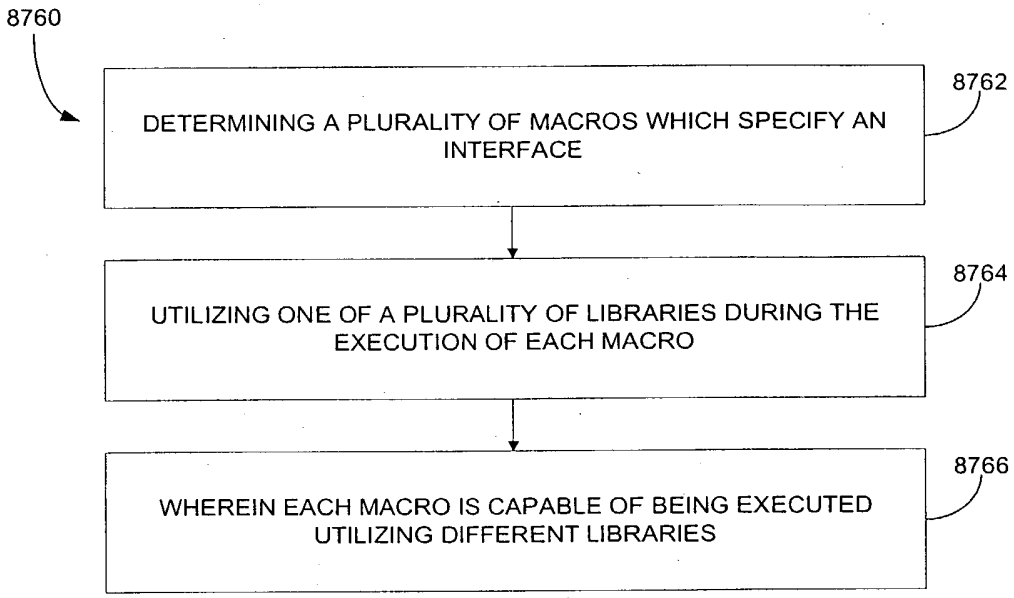

8760
DETERMINING A PLURALITY OF MACROS WHICH SPECIFY AN INTERFACE
8762
UTILIZING ONE OF A PLURALITY OF LIBRARIES DURING THE EXECUTION OF EACH MACRO
8764
WHEREIN EACH MACRO IS CAPABLE OF BEING EXECUTED UTILIZING DIFFERENT LIBRARIES
8766

Fig. 87C

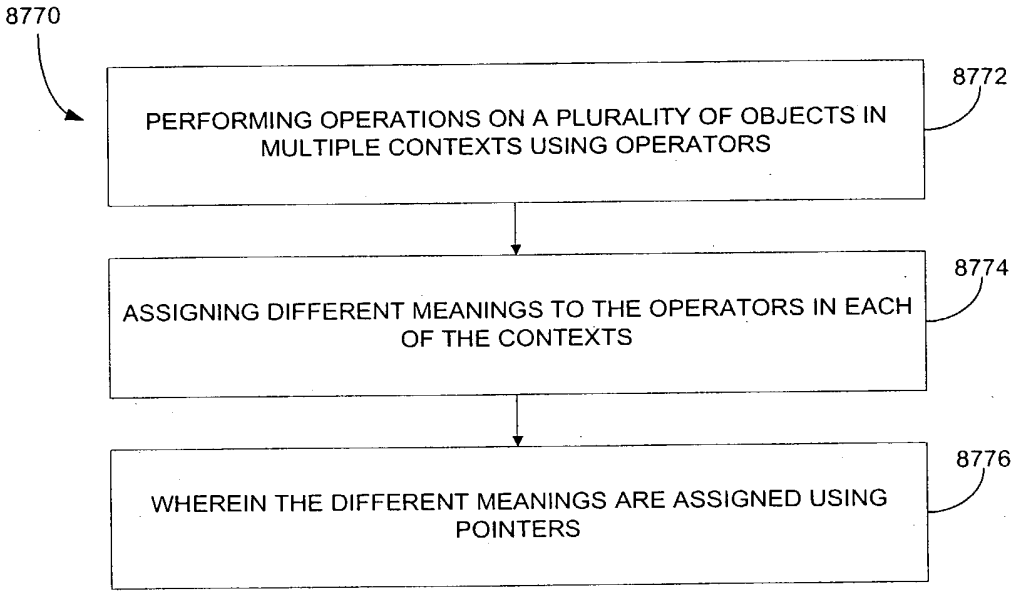

8770
PERFORMING OPERATIONS ON A PLURALITY OF OBJECTS IN MULTIPLE CONTEXTS USING OPERATORS
8772
ASSIGNING DIFFERENT MEANINGS TO THE OPERATORS IN EACH OF THE CONTEXTS
8774
WHEREIN THE DIFFERENT MEANINGS ARE ASSIGNED USING POINTERS
8776

Fig. 87D

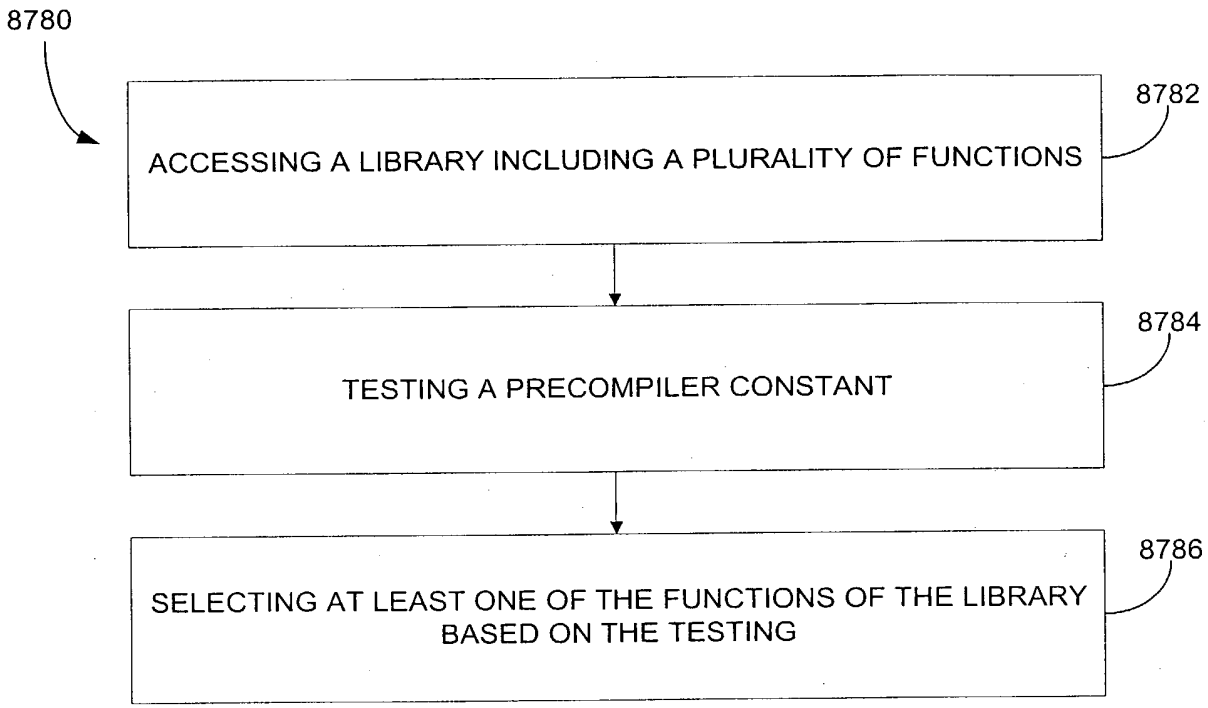

8780
ACCESSING A LIBRARY INCLUDING A PLURALITY OF FUNCTIONS
8782
TESTING A PRECOMPILER CONSTANT
8784
SELECTING AT LEAST ONE OF THE FUNCTIONS OF THE LIBRARY BASED ON THE TESTING
8786

Fig. 87E

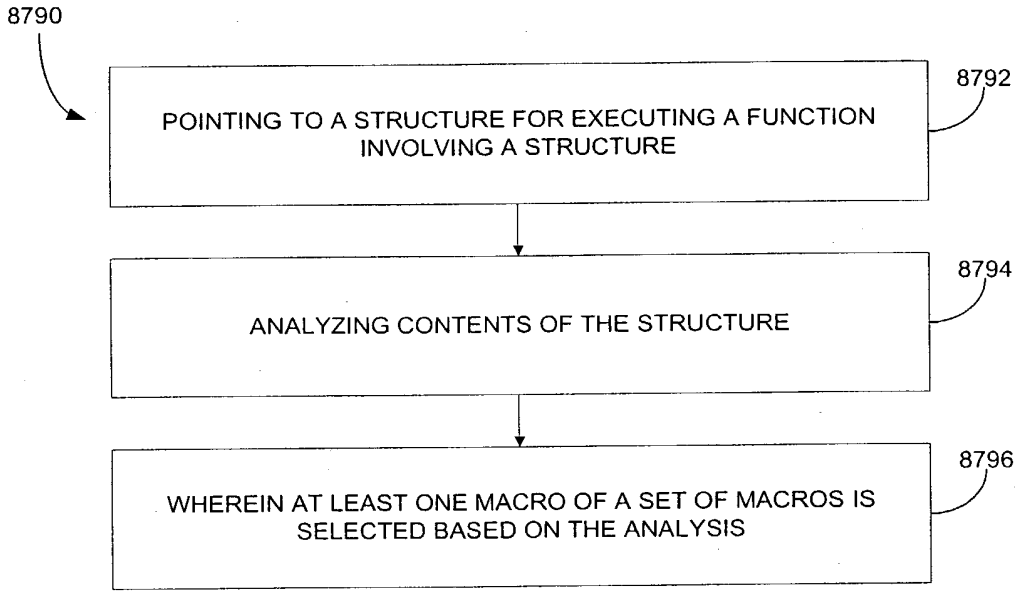

8790
POINTING TO A STRUCTURE FOR EXECUTING A FUNCTION INVOLVING A STRUCTURE
8792
ANALYZING CONTENTS OF THE STRUCTURE
8794
WHEREIN AT LEAST ONE MACRO OF A SET OF MACROS IS SELECTED BASED ON THE ANALYSIS
8796

Fig. 87F

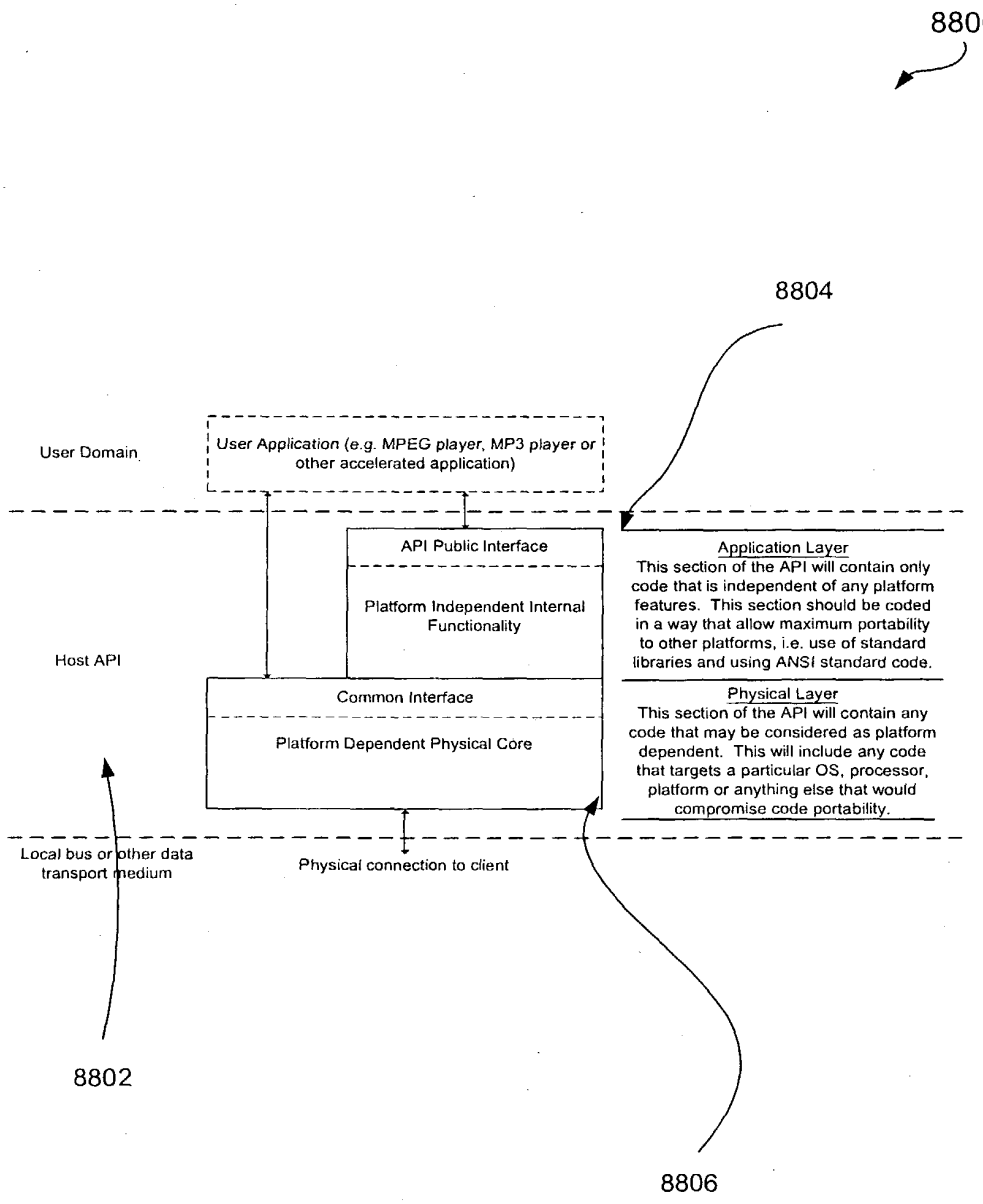

8800
8804
User Domain
User Application (e.g. MPEG player, MP3 player or other accelerated application)
API Public Interface
Platform Independent Internal Functionality
Host API
Common Interface
Platform Dependent Physical Core
Application Layer
This section of the API will contain only code that is independent of any platform features. This section should be coded in a way that allow maximum portability to other platforms, i.e. use of standard libraries and using ANSI standard code.
Physical Layer
This section of the API will contain any code that may be considered as platform dependent. This will include any code that targets a particular OS, processor, platform or anything else that would compromise code portability.
Local bus or other data transport medium
Physical connection to client
8802
8806

FIG. 88

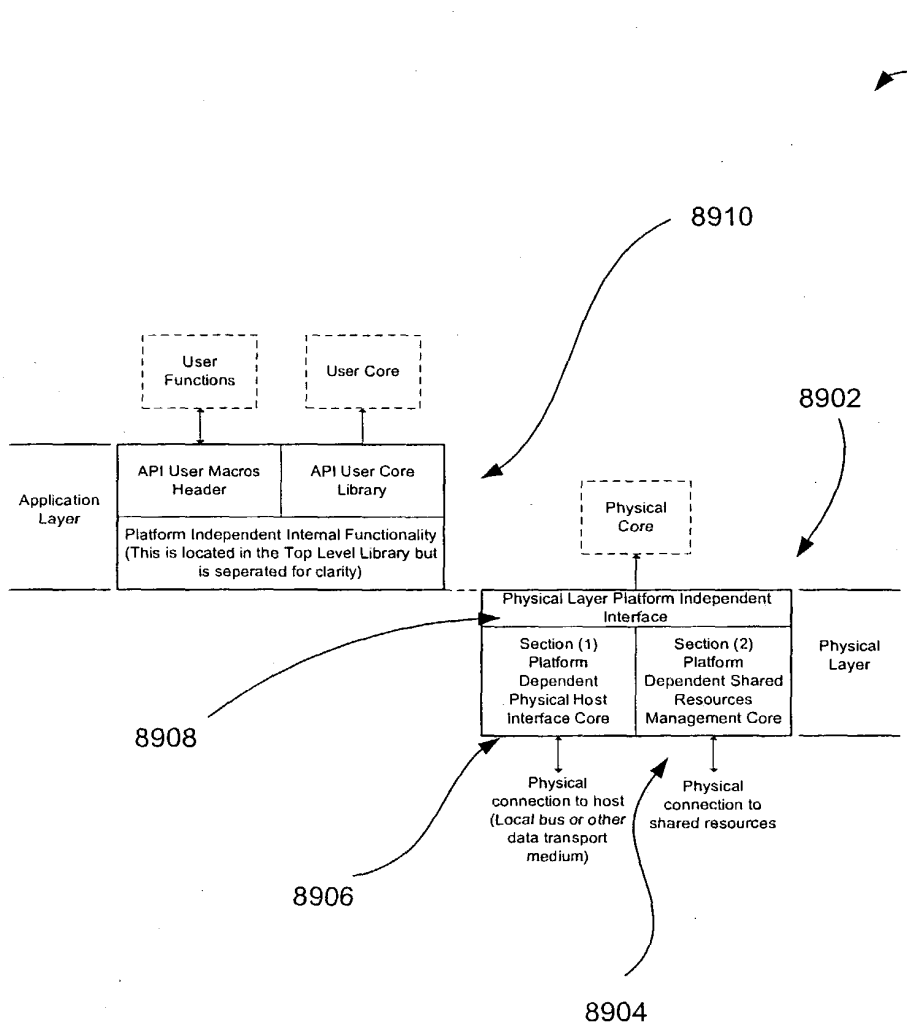

8900
8910
User Functions
User Core
8902
API User Macros Header
API User Core Library
Application Layer
Platform Independent Internal Functionality (This is located in the Top Level Library but is seperated for clarity)
Physical Core
Physical Layer Platform Independent Interface
Section (1) Platform Dependent Physical Host Interface Core
Section (2) Platform Dependent Shared Resources Management Core
Physical Layer
8908
Physical connection to host (Local bus or other data transport medium)
Physical connection to shared resources
8906
8904

FIG. 89

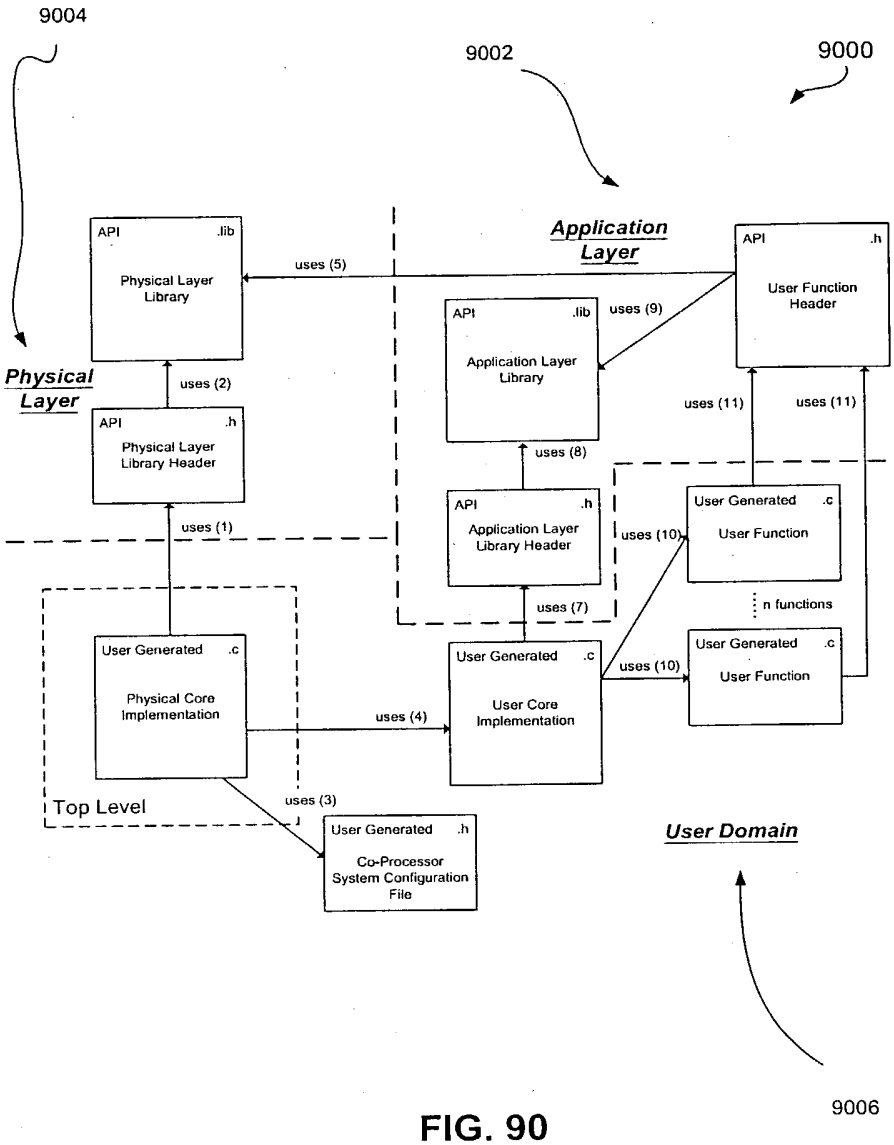

9004
9002
9000
Physical Layer
Application Layer
User Domain
API .lib
Physical Layer Library
API .h
Physical Layer Library Header
API .lib
Application Layer Library
API .h
Application Layer Library Header
API .h
User Function Header
User Generated .c
Physical Core Implementation
User Generated .c
User Core Implementation
User Generated .c
User Function
User Generated .c
User Function
n functions
User Generated .h
Co-Processor System Configuration File
Top Level
uses (1)
uses (2)
uses (3)
uses (4)
uses (5)
uses (7)
uses (8)
uses (9)
uses (10)
uses (10)
uses (11)
uses (11)
9006

FIG. 90

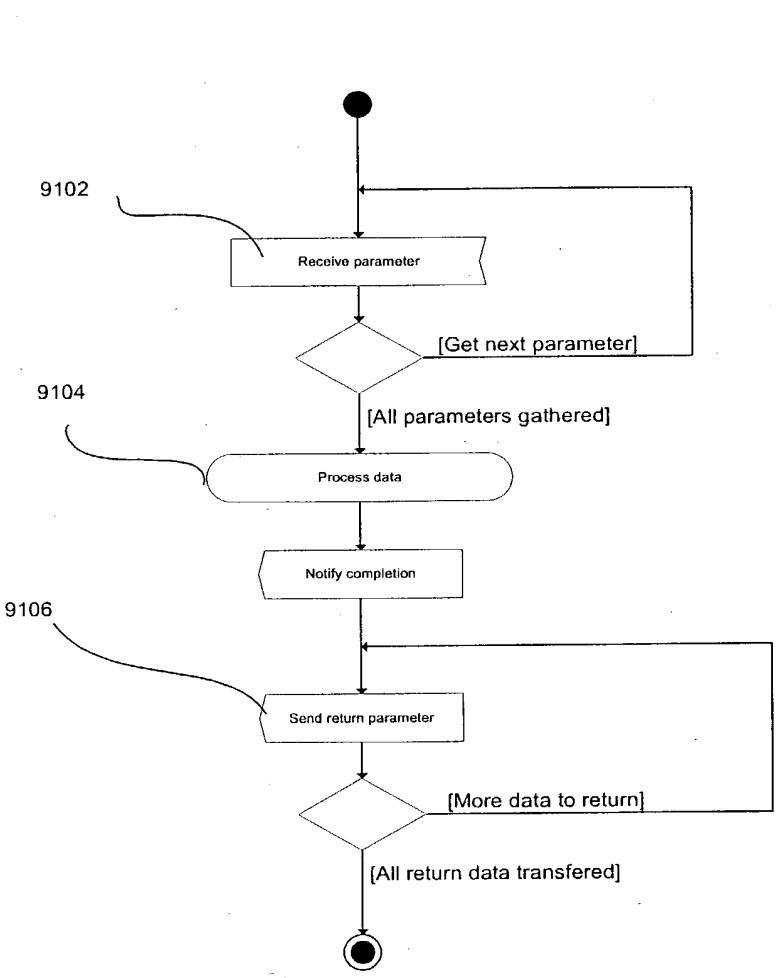

9100
9102
Receive parameter
[Get next parameter]
9104
[All parameters gathered]
Process data
Notify completion
9106
Send return parameter
[More data to return]
[All return data transfered]

FIG. 91

9200

Typical Address Packet

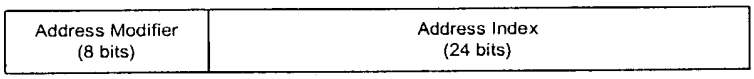



FIG. 92

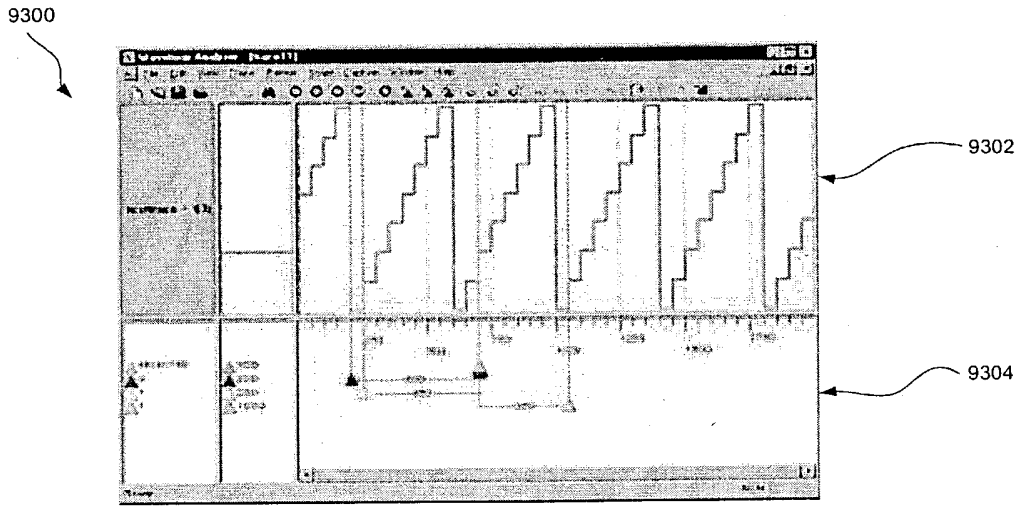



Fig. 93

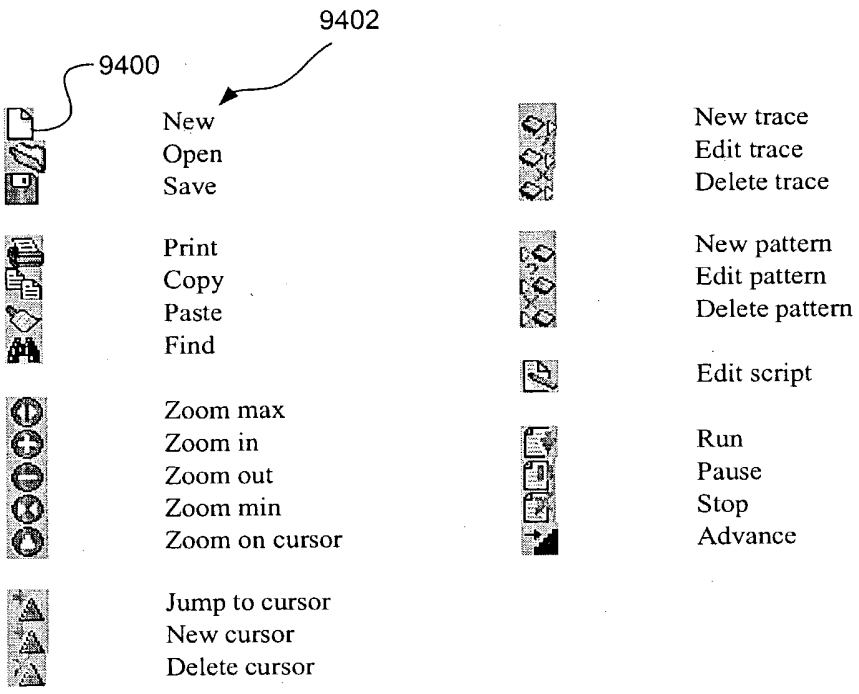

9402
9400
New
Open
Save
Print
Copy
Paste
Find
Zoom max
Zoom in
Zoom out
Zoom min
Zoom on cursor
Jump to cursor
New cursor
Delete cursor
New trace
Edit trace
Delete trace
New pattern
Edit pattern
Delete pattern
Edit script
Run
Pause
Stop
Advance

Fig. 94